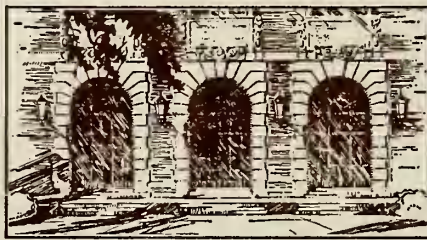


LIBRARY OF THE
UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

510.84
IL6r
no. 661-666
cop. 2



CENTRAL CIRCULATION BOOKSTACKS

The person charging this material is responsible for its renewal or its return to the library from which it was borrowed on or before the **Latest Date** stamped below. **You may be charged a minimum fee of \$75.00 for each lost book.**

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

TO RENEW CALL TELEPHONE CENTER, 333-8400

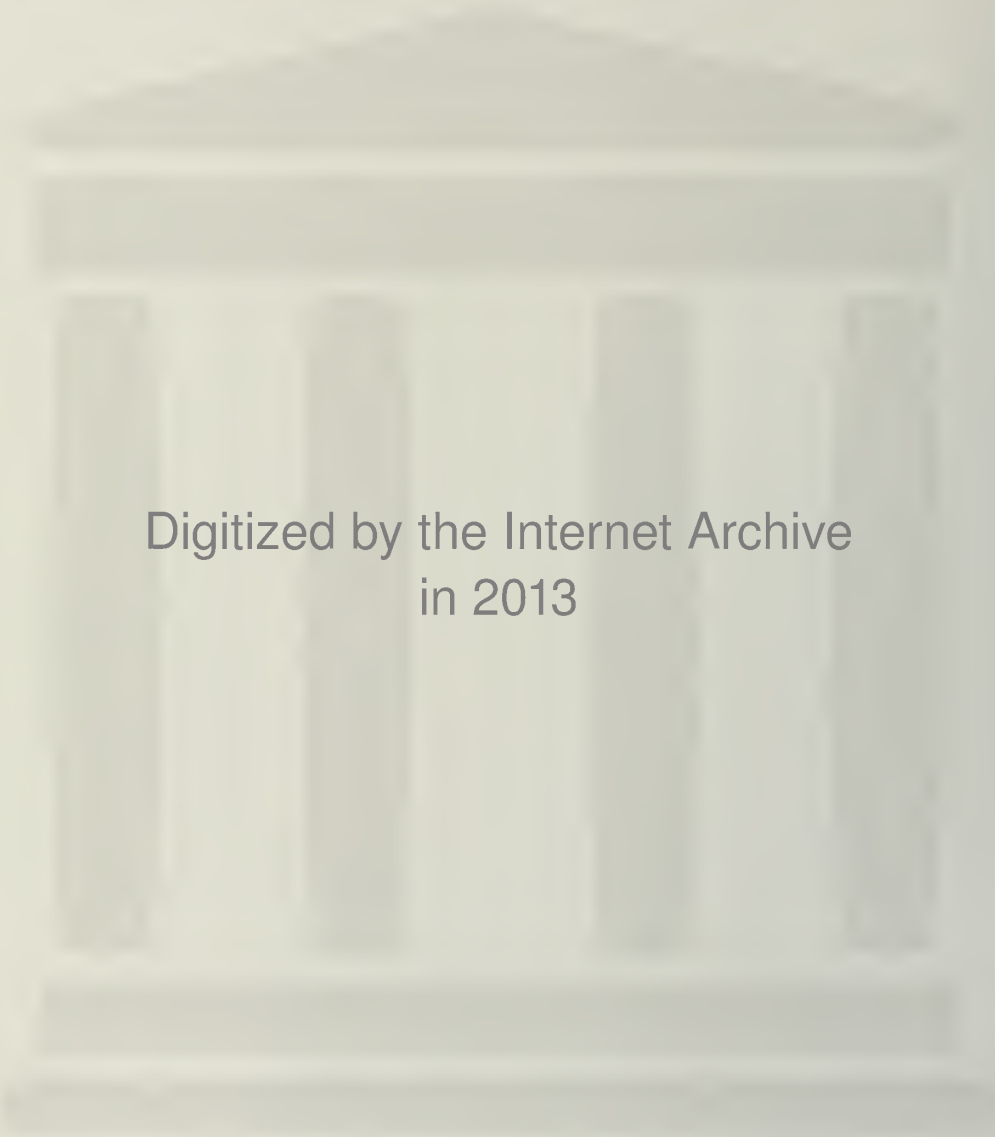
UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

AUG 14 1996

AUG 08 1996

When renewing by phone, write new due date below previous due date.

L162



Digitized by the Internet Archive
in 2013

<http://archive.org/details/designofnornetwo661legg>

170.84
Il6N
no 661
Report No. UIUCDCS-R-74-661

THE DESIGN OF NOR NETWORKS
UNDER FAN-IN AND FAN-OUT CONSTRAINTS
(A PROGRAMMING MANUAL FOR FIFOTRAN-G1)

by

Jeffrey Garnet Legge

June, 1974



THE LIBRARY OF THE

JUL 31 1974

UNIVERSITY OF ILLINOIS
AT URBANA-CHAMPAIGN

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

The person charging this material is responsible for its return to the library from which it was withdrawn on or before the **Latest Date** stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

SEP 6 1974
SEP - 9 RECD

13
Report No. UIUCDCS-R-74-661

THE DESIGN OF NOR NETWORKS UNDER FAN-IN AND FAN-OUT CONSTRAINTS*
(A PROGRAMMING MANUAL FOR FIFOTRAN-G1)

by

Jeffrey Garnet Legge

June, 1974

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

*This work was supported in part by the Department of Computer Science and submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, June 1974. This work was supported in part by the National Science Foundation under Grant No. NSF-GJ-40221.

510.84
Illor
no. 661-666
copy. 2

ACKNOWLEDGMENT

iii

The author would like to express his sincere gratitude to his thesis advisor, Professor S. Muroga, for his valuable suggestions and help in the preparation of this paper and more especially for his valuable time spent in the careful reading of this thesis.

The author would like to thank Mr. Jay Culliney for his guidance and his indispensable assistance throughout the period of this thesis research and Dr. Y. Kambayashi for many illuminating conversations relevant to this paper.

Finally, the author is grateful to his wife, Colleen, for her support and constant encouragement throughout the past months.

TABLE OF CONTENTS

	Page
1. INTRODUCTION	1
2. EXPLANATION OF TRANSFORMATIONS	6
3. PROGRAM ORGANIZATION	27
4. STATISTICS AND RESULTS	45
5. CONCLUSIONS.	64
APPENDIX.	68
LIST OF REFERENCES.	79

1. INTRODUCTION

The design of NOR (NAND) networks under fan-in, fan-out constraints is important from the practical point of view. Classical minimization procedures neglect the fan-in and fan-out constraints which are placed upon the logic designer. A simple algorithm which can find minimal NOR* networks while taking into account the fan-in and fan-out constraints of a NOR gate is the topic of this thesis.

Some of the published papers in this area that the author is aware of are [1], [2], and [4]. In [1], a computer oriented factoring algorithm generates a subset of common factors, selects optimum factors and indicates a realization for the factored equation. Only single-output, two-level networks with fan-in limited NOR gates are considered. In [2], fan-in limited NAND networks of intermediate to large size are considered. Some factoring techniques are used to construct tree networks with reduced gate count or levels of logic. In [4], a computer algorithm for the synthesis of multiple-output, multilevel NAND networks with fan-in and fan-out constraints is presented. Common factors with the largest figure-of-merit are sought and the network is transformed, if possible, in a manner such that both fan-in and fan-out problems can be resolved simultaneously. Sometimes cascaded NAND gates as shown in Fig. 1 are used to solve fan-in and fan-out problems separately.

Recently, the research group led by Professor S. Muroga (consisting of Y. Kambayashi, J. Culliney, H. Lai, and other students) of the

* Since NOR and NAND are dual functions, the algorithm can also be used for NAND logic.

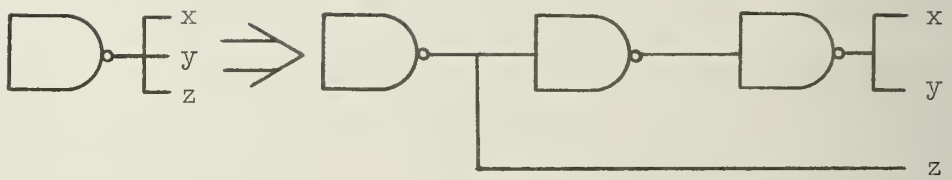
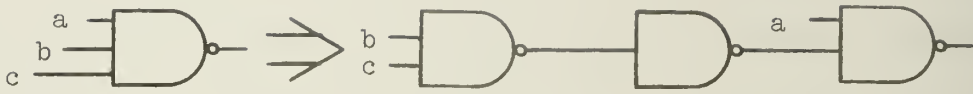


Fig. 1 Solution of fan-in and fan-out problems by using cascaded NAND gates.

Department of Computer Science at the University of Illinois at Urbana, Illinois, in an attempt to compromise between obtaining guaranteed minimal networks and expending excessive amounts of computation time, has developed some NOR network transformation procedures, called "transduction" (for: transformation and reduction) procedures. Starting with any "non-optimal" network (obtained, for example, by a very simple synthesis algorithm), these procedures are intended to obtain a "near-optimal" network. Fortunately, the transduction programs do not exhibit an exponential growth of computation time, as the number of gates involved in a problem increases. Thus, even problems involving many gates can be quickly solved.

In this paper, the capability of designing NOR networks under fan-in and fan-out constraints is combined with these transduction programs. The procedures for resolving fan-in and fan-out problems include the following transformations: (1) cascaded NOR gates; (2) the parallel duplication of a gate; (3) a composite transformation (resolution of both fan-in and fan-out problems simultaneously); and (4) additional more specialized transformations.

This paper presents a computer algorithm (contained in the program FIFOTRAN-G1) that will convert any multiple-output, multilevel NOR network that does not satisfy specified fan-in, fan-out constraints into a network which satisfies the constraints by applying appropriate transformations.

Detailed explanations of the individual transformations are given in Section 2 while descriptions of the program's (FIFOTRAN-G1's) organization and use are given in Section 3. Statistical results for 60 test functions run on an IBM 360/75 are given and discussed in Section 4.

In order to describe the transformations in Section 2 some notation and definitions must be introduced:

Let n be the number of (independent) external variables.

Let R be the number of (non-isolated) gates in the network.

Let the set of external variables be $x_1, x_2, \dots, x_n$ (and $\bar{x}_1, \bar{x}_2, \dots, \bar{x}_n$ when complements are to be available).

Let the set of gates in the network be $g_1, g_2, \dots, g_R$.

Let FI be the upper bound on the allowable fan-in to each gate of the network.

Let FOX be the upper bound on the allowable fan-out from each external variable (to other gates of the network).

Let FOO be the upper bound on the allowable fan-out from each output gate of the network.

Let FO be the upper bound on the allowable fan-out from each non-output gate of the network.

A gate g_j is said to be an immediate successor of gate g_i (or external variable x_i) if and only if g_j has g_i (or x_i) assigned as an input.

Let $IS(g_i)$ (or $IS(x_i)$) be the set of all immediate successors of the gate g_i (or external variable x_i).

A gate g_j is said to be an immediate predecessor of gate g_i if and only if $g_i \in IS(g_j)$.

Let $IP(g_i)$ be the set of all immediate predecessors of the gate g_i .

A gate g_i is said to have a fan-in problem if and only if $|IP(g_i)| > FI$.

An external variable x_i is said to have a fan-out problem if and only if $|IS(x_i)| > FOX$.

An output gate g_i is said to have a fan-out problem if and only if $|IS(g_i)| > FOO$.

A non-output gate g_i is said to have a fan-out problem if and only if $|IS(g_i)| > FO$.

The label $g_i^!$ (or $x_i^!$) indicates gate g_i (or x_i) after the set of immediate predecessors and/or immediate successors of g_i (or external variable x_i) has been changed as the result of the application of a particular transformation.

2. EXPLANATION OF TRANSFORMATIONS

Any NOR network that does not satisfy the specified fan-in, fan-out constraints can be changed by the network designer into a lowcost network which satisfies the constraints by using the transduction methods in combination with certain appropriate additional transformations. A set of six such transformations are proposed in this paper. Transformations T1, T2, and T3 are used to solve fan-out problems. Transformations T4 and T5 are used to solve fan-in problems. Transformation T6 is used to solve the composite problem—resolving fan-in and fan-out problems at the same time. This section explains the transformations and the conditions under which they are applied.

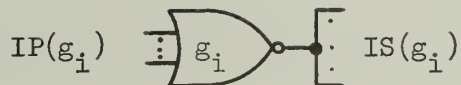
2.1 Transformations Used to Solve Fan-Out Problems

Transformations T1, T2, and T3 are used to solve fan-out problems. C1 and C2 are the sets of conditions which must be satisfied in order that T1 and T2 may be applied, respectively. T3 is applied if neither C1 nor C2 is satisfied (thus a set of conditions, C3, is not required for T3).

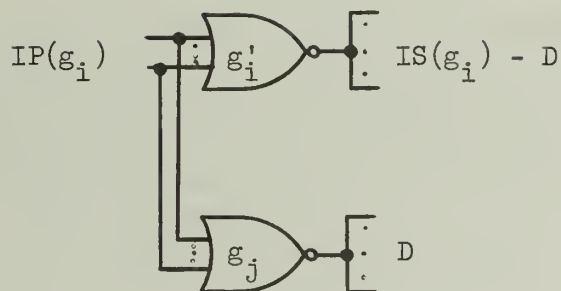
2.1.1 Transformation T1 (Parallel Duplication)

This transformation (parallel duplication) is used to solve the fan-out problem of any NOR gate which satisfies the condition set C1. T1 is implemented by subroutine TRANA.

When parallel duplication is used, only one gate is added to the network while the number of levels remains the same. For example, if non-output gate g_i in Fig. 2.1.1.1 (a) has a fan-out problem and condition set C1



(a)



(b)

Fig. 2.1.1.1 (a) Gate g_i with fan-out problem

(b) Gate g_i 's fan-out problem is solved by applying T1.

After the transformation, the following relations

hold. If $|IS(g_i)| \leq 2 \times FO$, then $|IS(g'_i) - D| = FO$

and $|D| \leq FO$. If $|IS(g_i)| > 2 \times FO$, then $|IS(g'_i) - D|$

$> FO$ and $|D| = FO$. Here D denotes the set of gates

fed by the transferred connections.

is satisfied, then transformation T1 is applied as follows: a gate, g_j , is added to the network. The immediate predecessors of g_i are connected as inputs to g_j . As many output connections as necessary to correct g_i 's fan-out problem—up to a maximum number equal to FO—are transferred from g_i to g_j (let the set of gates fed by transferred connections be designated D) (see Fig. 2.1.1.1 (b)). Thus, in the case when $|IS(g_i)|$ is originally greater than two times FO, g_i will still have a fan-out problem (although less than the original problem) following the transformation, and further transformations will have to be employed.

The case for an output gate g_i can be treated similarly.

2.1.2 Condition Set C1

The conditions for the application of T1 are checked in subroutine CONDA. The transformation T1 can only be used for gates and not for external variables. The conditions are listed below:

$$(1) \quad |IS(g_i)| > \begin{cases} FO & \text{if } g_i \text{ is a non-output gate} \\ FOO & \text{if } g_i \text{ is an output gate} \end{cases}$$

(2) The number of immediate predecessors of the gate g_i , $|IP(g_i)|$, must be less than or equal to the upper bound on the allowable fan-in of gate g_i (FI); i.e., $|IP(g_i)| \leq FI$. If this condition were not satisfied, then the new gate g_j that would be added to the network, when T1 is applied, would have a fan-in problem.

(3) For every gate $g_k \in IP(g_i)$: the number of immediate successors of the gate g_k , $|IS(g_k)|$, must be less than FO (or FOO for output gates). For any external variable $x_k \in IP(g_i)$: $|IS(x_k)|$ must be less than FOX.

2.1.3 Transformation T2

Transformation T2 is implemented by subroutine TRANE. T2 is a special transformation which is used to solve the fan-out problem of an external variable. When T2 is used, one gate is added to the network and the number of levels may be increased by at most one.

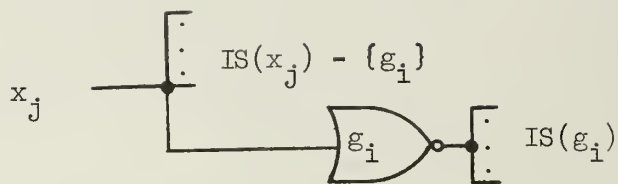
For example, if gate g_i in Fig. 2.1.3.1 has only one external variable, x_j , as an immediate predecessor and $|IS(x_j)|$ already exceeds the fan-out constraint FOX, then the fan-out problem of x_j can often be solved as follows: add a new gate, g_k , as an immediate successor to gate g_i and transfer as many output connections (excluding the connection to g_i) as necessary to correct x_j 's fan-out problem—up to a maximum number equal to FO—from x_j to the outputs of g_k (let the set of gates fed by the transferred connections be designated D). Then, the function x_j is available at the output of gate g_k . Thus, in the case when $|IS(x_j)|$ is originally greater than $FO + FOX$, x_j will still have a fan-out problem (although less than the original problem) following the transformation, and further transformations will have to be employed.

2.1.4 Condition Set C2

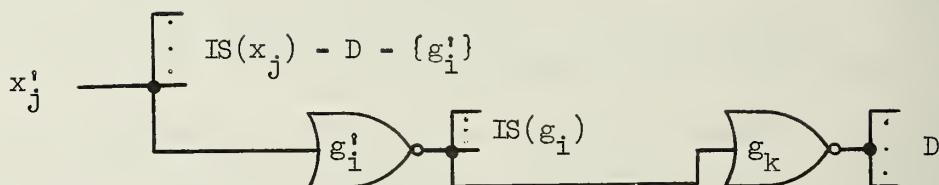
The conditions for the application of T2 are checked in subroutine CONDE. These conditions are shown below for an external variable, x_j , and an immediate successor, g_i .

$$(1) |IS(x_j)| > FOX$$

(2) There exists a gate g_i whose only immediate predecessor is external variable x_j : $IP(g_i) = \{x_j\}$



(a)



(b)

Fig. 2.1.3.1 (a) $IS(x_j)$ exceeds the fan-out constraint

(b) After application of T2, x_j has only $|IS(x_j) - D|$ immediate successors. However, if $|IS(x_j)| \leq FO + FOX$ originally, then $|IS(x_j) - D| = FOX$ and $|D| \leq FO$ after the transformation. If $|IS(x_j)| > FO + FOX$ originally, then $|IS(x_j) - D| > FOX$ and $|D| = FO$ after the transformation.

(3) For this particular gate g_i ,

$$|IS(g_i)| < \begin{cases} FO & \text{if } g_i \text{ is a non-output gate} \\ FOO & \text{if } g_i \text{ is an output gate} \end{cases}$$

2.1.5 Transformation T3

This transformation is used to solve the fan-out problem of a gate g_i (or external variable x_i) when neither condition set C1 nor condition set C2 is satisfied. T3 is implemented by subroutine TRANCA. When T3 is used, at least two gates are added to the network and the number of levels may be increased by at most two.

There are three general cases (each case will be discussed for a non-output gate, g_i , but the transformations for output gates and external variables are similar):

(1) If gate g_i has a fan-out problem and $FO < |IS(g_i)| \leq (2 \times FO) - 1$, then the corresponding transformation results in Fig. 2.1.5.1. An output from gate g_i is connected to a new gate, g_j . An output from g_j in turn is connected to another new gate, g_k . As many output connections as necessary to correct g_i 's fan-out problem are transferred from g_i to g_k (let the set of gates fed by the transferred connections be designated D).

(2) If gate g_i has a fan-out problem and $(2 \times FO) - 1 < |IS(g_i)| \leq FO^2 + FO - 1$, then the corresponding transformation results in Fig. 2.1.5.2. The only difference between this transformation and that shown in Fig. 2.1.5.1 is that gate g_j now fans out to ℓ newly added gates, $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$ where ℓ is the smallest integer such that $(\ell + 1) \times FO - 1 \geq |IS(g_i)|$. Output connections are transferred from g_i to $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$ such that $g_i, g_{k_1}, g_{k_2}, \dots, g_{k_{\ell-1}}$ each will have a fan-out of FO

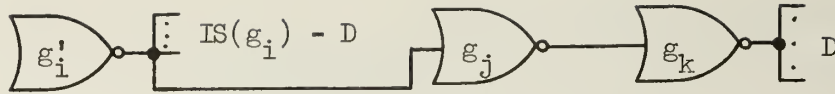


Fig. 2.1.5.1 Gate g_i after application of T3 when $FO < |IS(g_i)| \leq (2 \times FO) - 1$. After the transformation: $|IS(g_i) - D + 1| = FO$ and $|D| \leq FO$.

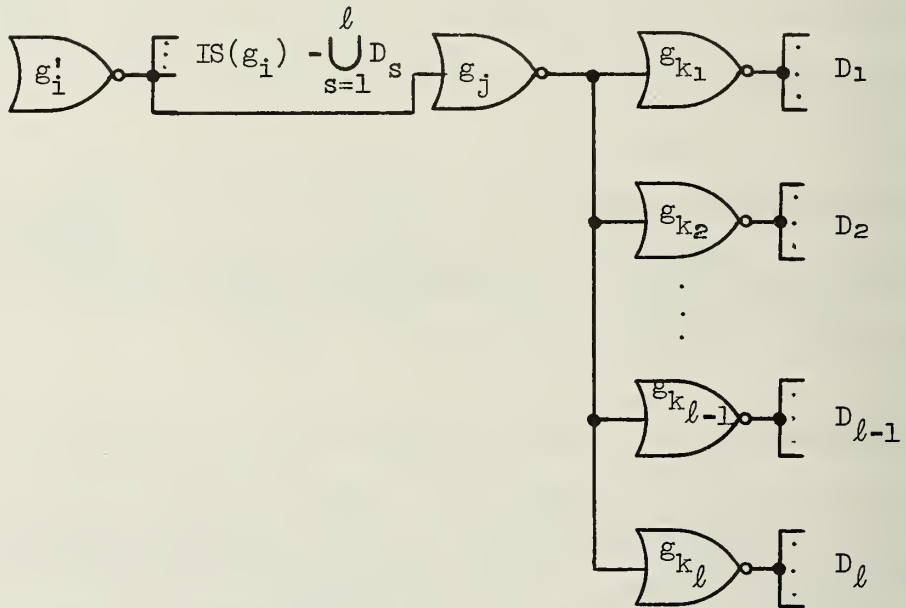


Fig. 2.1.5.2 Gate g_i after application of T3 when $(2 \times FO) - 1 < |IS(g_i)| \leq FO^2 + FO - 1$. After the transformation: $|D_1| = |D_2| = \dots = |D_{l-1}| = FO$, $|D_l| = |IS(g_i)| - (l \times FO) + 1 \leq FO$, $|IS(g'_i)| = |IS(g_i) - \bigcup_{s=1}^l D_s| + 1 = FO$, and $l \leq FO$.

while g_{k_ℓ} will have a fan-out of $|IS(g_i)| - \ell \times FO + 1$ (which will be $\leq FO$). Let the sets of connections transferred to $g_{k_1}, \dots, g_{k_\ell}$ be designated by $D_1, \dots, D_\ell$, respectively.

(3) If gate g_i has a fan-out problem and $|IS(g_i)| > FO^2 + FO - 1$ then the corresponding transformation results in Fig. 2.1.5.3. The only difference between this transformation and that described in Fig. 2.1.5.2 is that gate g_j now fans out to ℓ newly added gates, $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$, where ℓ is equal to FO . Output connections are transferred from g_i to $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$, such that $g_j, g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$ each will have a fan-out of FO ($|D_1| = |D_2| = \dots = |D_\ell| = FO$). In this case, g_i will still have a fan-out problem (although less than the original problem) following the transformation, and further transformations will have to be employed.

2.2 Transformations Used to Solve Fan-In Problems

Transformations T^4 and T^5 are used to solve fan-in problems. Condition set C^4 must be satisfied in order that T^4 may be applied. If C^4 is not satisfied, then transformation T^5 is used (thus, a set of conditions, C^5 , is not required for T^5).

2.2.1 Transformation T^4

Parallel duplication can not be used for solving fan-in problems, but with the help of existing gates it is sometimes possible to solve the fan-in problem of a gate.

For example, consider the two gates shown in Fig. 2.2.1.1 in which gate g_i has a fan-in problem and there exists a gate g_j such that $IP(g_j) \subset IP(g_i)$. If condition set C^4 is satisfied, then a new gate g_k is added as shown in

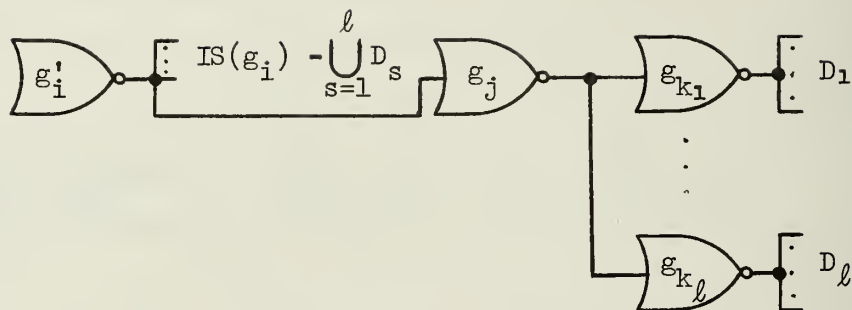


Fig. 2.1.5.3 Gate g_i after application of T3 when $|IS(g_i)| > FO^2 + FO - 1$. After the transformation: $|D_1| = |D_2| = \dots = |D_\ell|$, $\ell = FO$ and $|IS(g_i) - \bigcup_{s=1}^{\ell} D_s| + 1 > FO$.

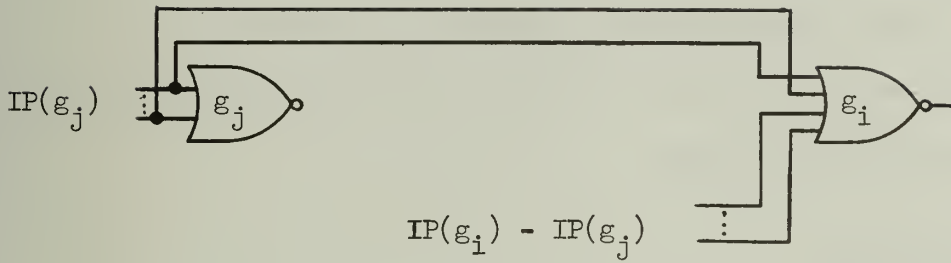


Fig. 2.2.1.1 Gate g_i has a fan-in problem and $IP(g_j) \subset IP(g_i)$ holds.

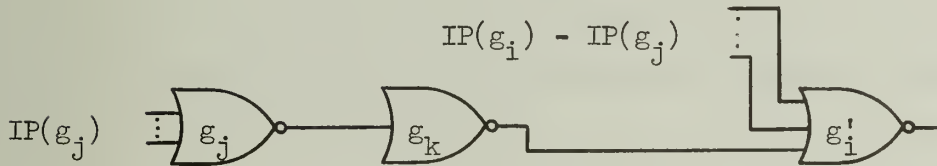


Fig. 2.2.1.2 The fan-in problem of g_i is solved by transformation T^4 .

After the transformation: $|IP(g_i) - IP(g_j)| + 1 \leq FI$.

Fig. 2.2.1.2. Gate g_k 's output is connected as an input to gate g_i , and the output of gate g_j is connected as an input to gate g_k . Finally, every connection from an immediate predecessor of g_j is removed from g_i .

When T4 is used, only one gate is added to the network while the increase in the number of levels is at most two. This transformation is implemented in subroutine TRANDB.

2.2.2 Condition Set C4

The conditions for application of T4 are checked in subroutine CONDDb. These conditions are listed below:

- (1) $|IP(g_i)| > FI$
- (2) There exists a gate g_j such that:
 - (a) $|IS(g_j)| < \begin{cases} FO & \text{if } g_j \text{ is a non-output gate} \\ FOO & \text{if } g_j \text{ is an output gate} \end{cases}$
 - (b) every immediate predecessor of g_j is also an immediate predecessor of g_i .
- (3) $|IP(g_i)| - |IP(g_j)| < FI$

2.2.3 Transformation T5

This transformation is used to solve the fan-in problem of a gate g_i when T4 can not be applied. T5 is implemented by subroutine TRANBB. When T5 is used, at least two gates are added to the network and the number of levels may be increased by at most two.

There are three general cases:

- (1) If gate g_i has a fan-in problem and $FI < |IP(g_i)| \leq (2 \times FI) - 1$, then the corresponding transformation results in Fig. 2.2.3.1. An output

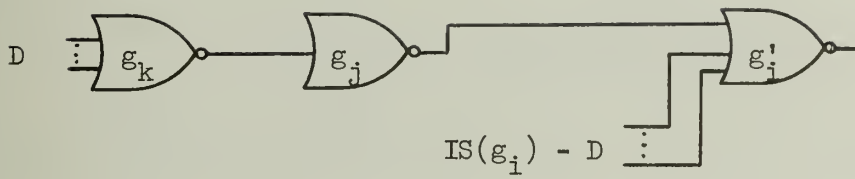


Fig. 2.2.3.1 Gate g_i after application of T5 when $FI < |IP(g_i)| \leq (2 \times FI) - 1$. After the transformation: $|IS(g_i) - D + 1| = FI$ and $|D| \leq FI$.

from a new gate, g_j , is first connected to g_i . As many input connections as necessary to correct g_i 's fan-in problem are transferred from g_i to a new gate g_k and then the output of gate g_k is connected to g_j (in Fig. 2.2.3.1, D designates the set of gates whose output connections are transferred).

(2) If gate g_i has a fan-in problem and $(2 \times FI) - 1 < |IP(g_i)| \leq FI^2$, then the corresponding transformation results in Fig. 2.2.3.2. This transformation and that shown in Fig. 2.2.3.1 are similar, but in this case gate g_i is fed by ℓ new gates $g_{j_1}, g_{j_2}, \dots, g_{j_\ell}$ where ℓ is the smallest integer such that $\ell \times (FI - 1) + FI \geq |IP(g_i)|$. For each gate g_{j_p} ($p = 1, 2, \dots, \ell$), the output of another new gate, g_{k_p} is connected to g_{j_p} . Input connections are transferred from g_i to $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$ such that $g_i, g_{k_1}, g_{k_2}, \dots, g_{k_{\ell-1}}$ each will have a fan-in of FI while g_{k_ℓ} will have a fan-in of $|IP(g_i)| - 1 \times (FI - 1)$ (which will be $\leq FI$). Here the set of connections transferred from g_i to g_{k_s} ($s = 1, \dots, \ell$) is designated D_s ($s = 1, \dots, \ell$), respectively.

(3) If gate g_i has a fan-in problem and $|IP(g_i)| > FI^2$, then the corresponding transformation results in Fig. 2.2.3.3. The only difference between this transformation and that shown in Fig. 2.2.3.2 is that gate g_i will still be left with a fan-in problem after the transformation. Outputs from ℓ new gates $g_{j_1}, g_{j_2}, \dots, g_{j_\ell}$, where $\ell = FI$, are connected to g_i . Again, for each gate g_{j_p} ($p = 1, 2, \dots, \ell$), the output of another new gate, g_{k_p} , is connected to g_{j_p} . Input connections are transferred from g_i to $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$ such that $g_{k_1}, g_{k_2}, \dots, g_{k_\ell}$ each will have a fan-in of FI . In this case, g_i will still have a fan-in problem (although less

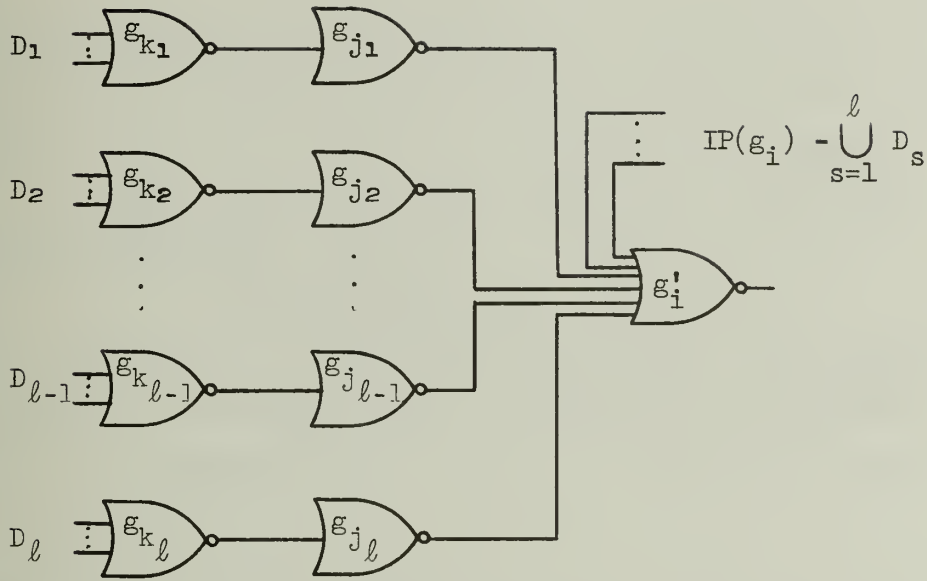


Fig. 2.2.3.2 Gate g'_i after application of T5 when $(2 \times FI) - 1 < |IP(g_i)| \leq FI^2$. After the transformation: $|D_1| = |D_2| = \dots = |D_{l-1}| = FI$, $|D_l| = |IP(g_i)| - l \times (FI - 1) \leq FI$, $|IP(g'_i)| = |IP(g_i) - \bigcup_{s=1}^l D_s| + l = FI$, and $l \leq FI$.

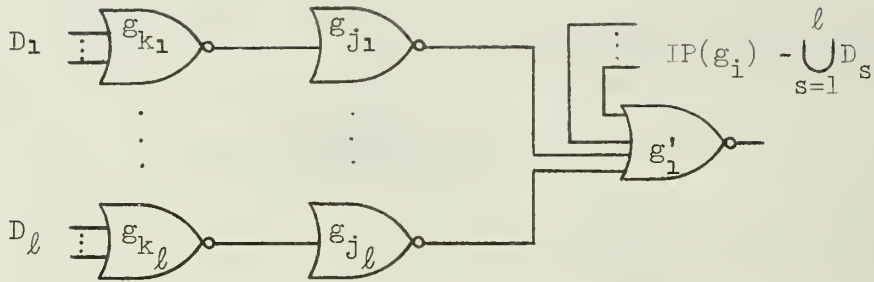


Fig. 2.2.3.3 Gate g_i after application of T5 when $|IP(g_i)| > FI^2$.

After the transformation: $|D_1| = |D_2| = \dots = |D_\ell| = FI$,

$$\ell = FI \text{ and } |IP(g'_i)| = |IP(g_i) - \bigcup_{s=1}^{\ell} D_s| + \ell = |IP(g_i)| - FI^2 + FI > FI.$$

severe than the original problem) following the transformation and further transformations will have to be employed.

2.3 The Composite Problem

Transformation T6 is used to solve both fan-in and fan-out problems at the same time. Condition set C6 must be satisfied in order that T6 may be applied.

2.3.1 Transformation T6

This transformation is applied when condition set C6 is satisfied. T6 is implemented by subroutine TRANCC. When T6 is used, at least two gates are added to the network while the number of levels may be increased by at most two.

For example, consider the original subnetwork configuration in Fig. 2.3.1.1 in which there exists a set of gates $K = \{ g_{i_1}, g_{i_2}, \dots, g_{i_k} \}$ and a set of gates $H = \{ g_{j_1}, g_{j_2}, \dots, g_{j_h} \}$ such that the output of each gate g_{i_1} is connected to each gate g_{j_1} . There are two general cases:

(1) If the subnetwork in Fig. 2.3.1.1 exists and satisfies condition set C6 and $h \leq FO$, then the corresponding transformation results in Fig.

2.3.1.2. An output from $g_{i_1}, g_{i_2}, \dots, g_{i_k}$ is connected to a new gate g_p . The output of g_p in turn is connected to another new gate g_q . Gate g_q then is connected as an input to $g_{j_1}, g_{j_2}, \dots, g_{j_h}$. The h output connections from each of gates $g_{i_1}, g_{i_2}, \dots, g_{i_k}$ to gates $g_{j_1}, g_{j_2}, \dots, g_{j_h}$ are removed.

(2) If the subnetwork in Fig. 2.3.1.1 exists and satisfies condition set C6 and $FO \leq h \leq FO^2$, then the corresponding transformation results in

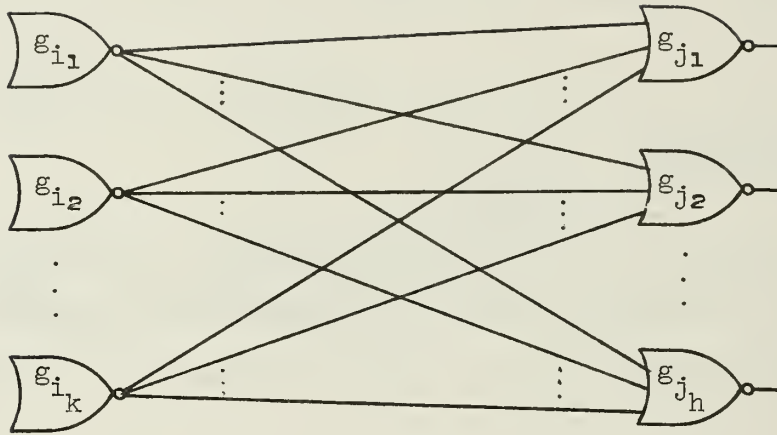


Fig. 2.3.1.1 Subnetwork structure upon which transformation T_6 may be applied.

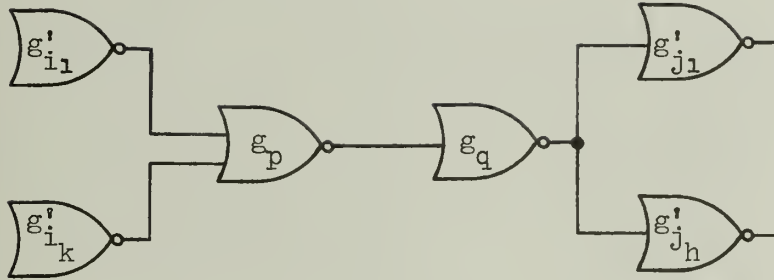


Fig. 2.3.1.2 Subnetwork structure (Fig. 2.3.1.1) after the application of T6 when $h \leq FO$. After the transformation: $|IS(g'_{i_1})| = |IS(g_{i_1})| + 1 - h$, $|IS(g'_{i_2})| = |IS(g_{i_2})| + 1 - h$, $\dots$, $|IS(g'_{i_k})| = |IS(g_{i_k})| + 1 - h$. $|IP(g'_j)| = |IP(g_j)| + 1 - k$, $|IP(g'_{j_2})| = |IP(g_{j_2})| + 1 - k$, $\dots$, $|IP(g'_{j_h})| = |IP(g_{j_h})| + 1 - k$.

Fig. 2.3.1.3. The only difference between this transformation and that shown in Fig. 2.3.1.2 is that gate g_p now fans out to ℓ newly added gates $g_{q_1}, g_{q_2}, \dots, g_{q_\ell}$ where ℓ is the smallest integer such that $\ell \times FO \geq h$. Output connections from each of $g_{q_1}, g_{q_2}, \dots, g_{q_{\ell-1}}$ will be fed to FO of the g_j 's (i.e. g_{q_1} will feed $g_{j_1}, g_{j_2}, \dots, g_{j_{FO}}$; g_{q_2} will feed $g_{j_{FO+1}}, g_{j_{FO+2}}, \dots, g_{j_{2FO}}$; etc.). Gate g_{q_ℓ} will fan-out to the $h - (\ell - 1) FO$ gates: $g_{j_{(\ell-1)FO+1}}, \dots, g_{j_h}$. The h output connections from each of gates $g_{i_1}, g_{i_2}, \dots, g_{i_k}$ to gates $g_{j_1}, g_{j_2}, \dots, g_{j_k}$ are removed.

After T6 is applied at least one of the gates in Fig. 2.3.1.1 has its fan-in or fan-out problem completely solved while the fan-in and fan-out problems of the rest of the gates may only be partially solved.

2.3.2 Condition Set C6

The conditions for the application of T6 are checked in subroutine CONDCC. A subnetwork of gates (or gates and external variables) having the generalized structure shown in Fig. 2.3.1.1 is sought which satisfies the following conditions:

(1) There exists a set of gates and/or external variables $K = \{ x_{i_1}, \dots, x_{i_t}, g_{i_t+1}, \dots, g_{i_k} \}$ (t may be 0 or t may equal k in some cases) and a set of gates $H = \{ g_{j_1}, g_{j_2}, \dots, g_{j_h} \}$ such that for every $g_{j_r} \in H$, $IP(g_{j_r}) \supseteq K$ holds, for every $g_{i_s} \in K$, $IS(g_{i_s}) \supseteq H$ holds, and for every $x_{i_r} \in K$, $IS(x_{i_r}) \supseteq H$ holds.

(2) $2 \leq k \leq FI$.

(3) $2 \leq h \leq FO^2$.

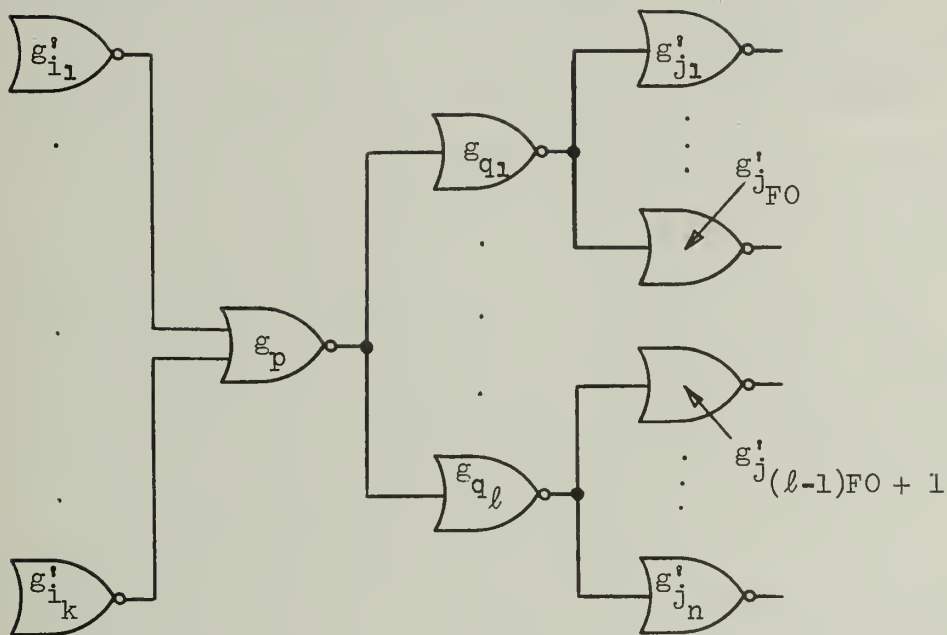


Fig. 2.3.1.3 Subnetwork structure (Fig. 2.3.1.1) after the application of T6 when $FO < h \leq FO^2$. After the transformation:

$$\begin{aligned}
 |IS(g_{i_1}')| &= |IS(g_{i_1})| + 1 - h, \quad |IS(g_{i_2}')| = |IS(g_{i_2})| + \\
 &1 - h, \quad \dots, \quad |IS(g_{i_k}')| = |IS(g_{i_k})| + 1 - h. \quad |IP(g_{j_1}')| = \\
 &|IP(g_{j_1})| + 1 - k, \quad |IP(g_{j_2}')| = |IP(g_{j_2})| + 1 - k, \quad \dots, \\
 &|IP(g_{j_h}')| = |IP(g_{j_h})| + 1 - k.
 \end{aligned}$$

(4) The number of gates in K with fan-out problems plus the number of gates in H with fan-in problems must be at least 2.

(5) At least one fan-in or fan-out problem must be completely resolved by the transformation.

3. PROGRAM ORGANIZATION

This program consists of two major parts controlled by a main program (referred to as FIFOTRAN-G1). The first part (Part A) consists of 9 subroutines, MAIN, CONCCO, ELANDO, MINI2, NORNET, OUTPUT, PROCII, RNONES, SUBNET, (written by Culliney, Lai, and Kambayashi) which read in a given function or set of functions, synthesize an initial network to realize the given function(s) using a simple algorithm, and attempt to minimize the number of gates without regard to fan-in and fan-out constraints. The second part (Part B), consisting of 16 subroutines, takes the "reduced" network derived by Part A and solves the fan-in and fan-out problems of all gates that exceed the constraints specified by the user.

All subroutines were written in FORTRAN IV, compiled by the FORTRAN H (opt 2) compiler, and executed on the IBM 360/75J.

3.1 Flowchart Description of Entire Program

The general organization and linkages between the main program and the subroutines of Part A and Part B are shown in the flow diagram in Fig. 3.1.1, and the functions of the various blocks are described as follows:

(1) The given functions to be realized and constraints are read in by the program.

(2) The subroutine NORNET (programmed by Y. Kambayashi) synthesizes the original network to realize the given function (s). There are many other subroutines that can be used to synthesize the original network but none of them were used in FIFOTRAN-G1.

(3) Subroutine PROCII (programmed by J. Culliney) is executed repeatedly

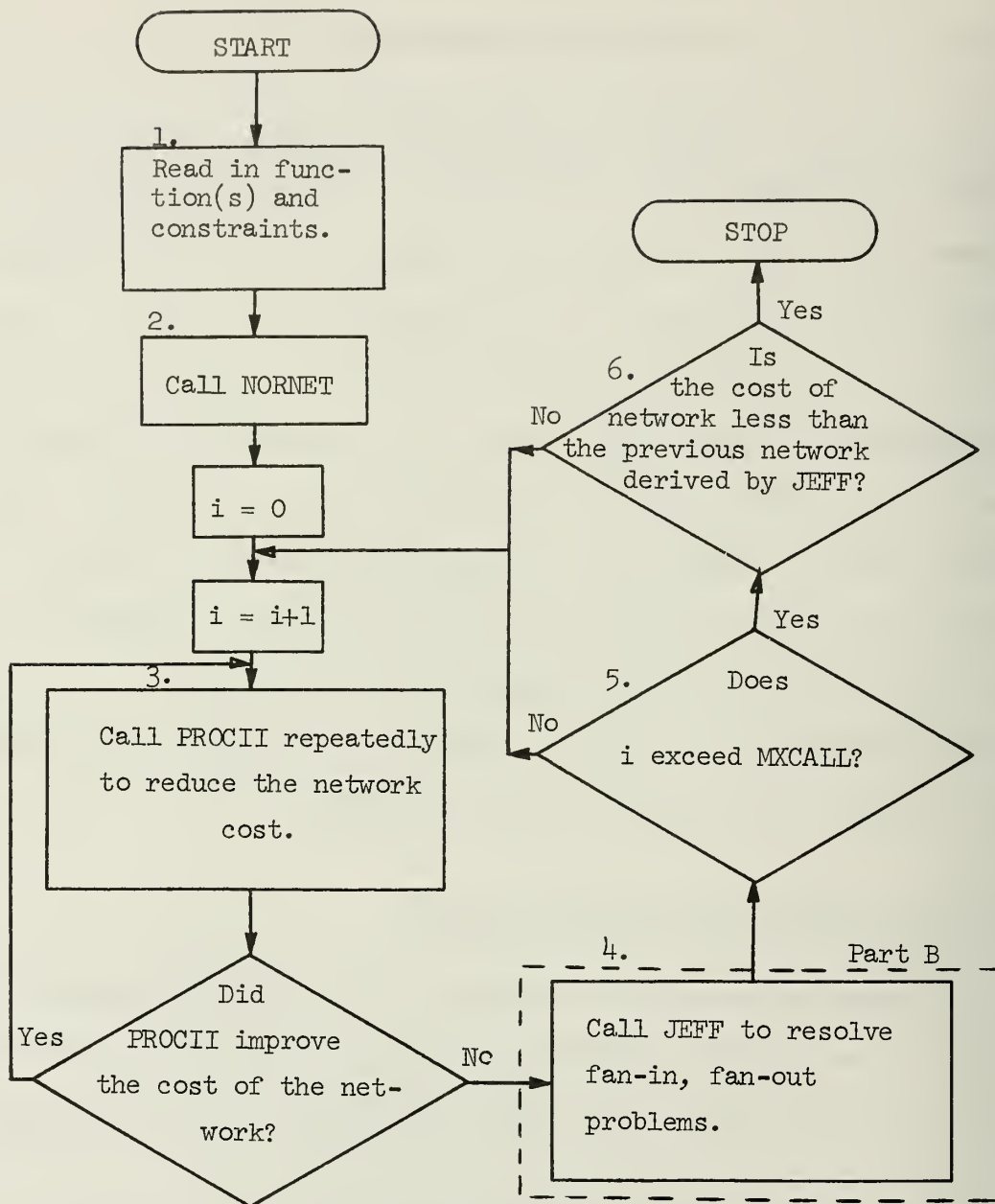


Fig. 3.1.1 General program flow diagram.

to reduce the cost of the current network, without regard to fan-in, fan-out constraints, until no further net savings (i.e., reduction in cost) can be obtained. The cost of the network is defined as follows:

$$\text{Cost} = A \times (\text{no. of gates in network}) + B \times (\text{no. of interconnections})$$

where A and B are positive integral weights. In this program, $A = 100$ and $B = 1$.

(4) Subroutine JEFF solves the fan-in and fan-out problems of the network, attempting to add as few additional gates as possible to the network.

(5) If the number of iterations of the entire fan-in, fan-out solving procedure is less than MXCALL, then the program returns to (3), otherwise it continues to (6). MXCALL is the lower limit on the number of iterations which the program in Fig. 3.1.1 is required to execute. In other words, the program is required to iterate MXCALL times regardless of the network cost. After the procedure has iterated MXCALL times, the program continues to iterate until there is no further reduction in the cost of the network. The parameter MXCALL is easily adjusted by the programmer.

(6) If the cost of the current network derived by JEFF is less than the cost of the previous network derived by JEFF, then return to step (3), otherwise stop.

3.2 Subroutine Descriptions and Flowchart for Part B of the Program

The controlling subroutine in Part B is subroutine JEFF. Part B consists of 16 subroutines: CHGATE, CONDA, CONDCC, CONDDb, CONDE, FDGATE, FDSUB, JEFF, PSORT, SSORT, TRANA, TRANBB, TRANCA, TRANCC, TRANDB, and TRANE. A brief description of each of these subroutines is given below.

- CHGATE: This subroutine updates certain necessary parameters whenever subroutine JEFF adds new gates to the network.
- CONDA, CONDCC, CONDDDB, CONDE: These subroutines test condition sets C_1 , C_6 , C_4 , and C_2 respectively. (see Section 2)
- FDGATE: This subroutine constructs a list of available gates, i.e., all the "isolated gates" (i.e., unused gates which have been disconnected from previously existing networks) which have not been used in the network most currently derived by PROCII. This list is constructed every time block 4 in Fig. 3.1.1 is executed. A maximum of 80 gates plus external variables is allowed.
- FDSUB: This subroutine locates all gates with fan-in and fan-out problems. This subroutine and CONDCC are used to select the particular fan-in or fan-out problem to be solved next.
- JEFF: This subroutine controls the general order of execution of most of the other subroutines. All error messages and printing are generated in this subroutine. A flowchart of JEFF is shown in Fig. 3.2.1 and is explained later in this section.
- PSORT: This subroutine sorts the immediate predecessors of gate g_i according to level number and puts them in descending order. For example, the network in Fig. 3.2.2 illustrates the numbering of levels with the output gate(s) in level one. For $g_i = g_1$ in this network, PSORT would construct a list which looks like that in Fig. 3.2.3.
- SSORT: This subroutine sorts the immediate successors of gate g_i according to level number and puts them in ascending order.
- TRANA: This subroutine implements transformation T1 which is used to solve the fan-out problem of a specified gate by parallel duplication of

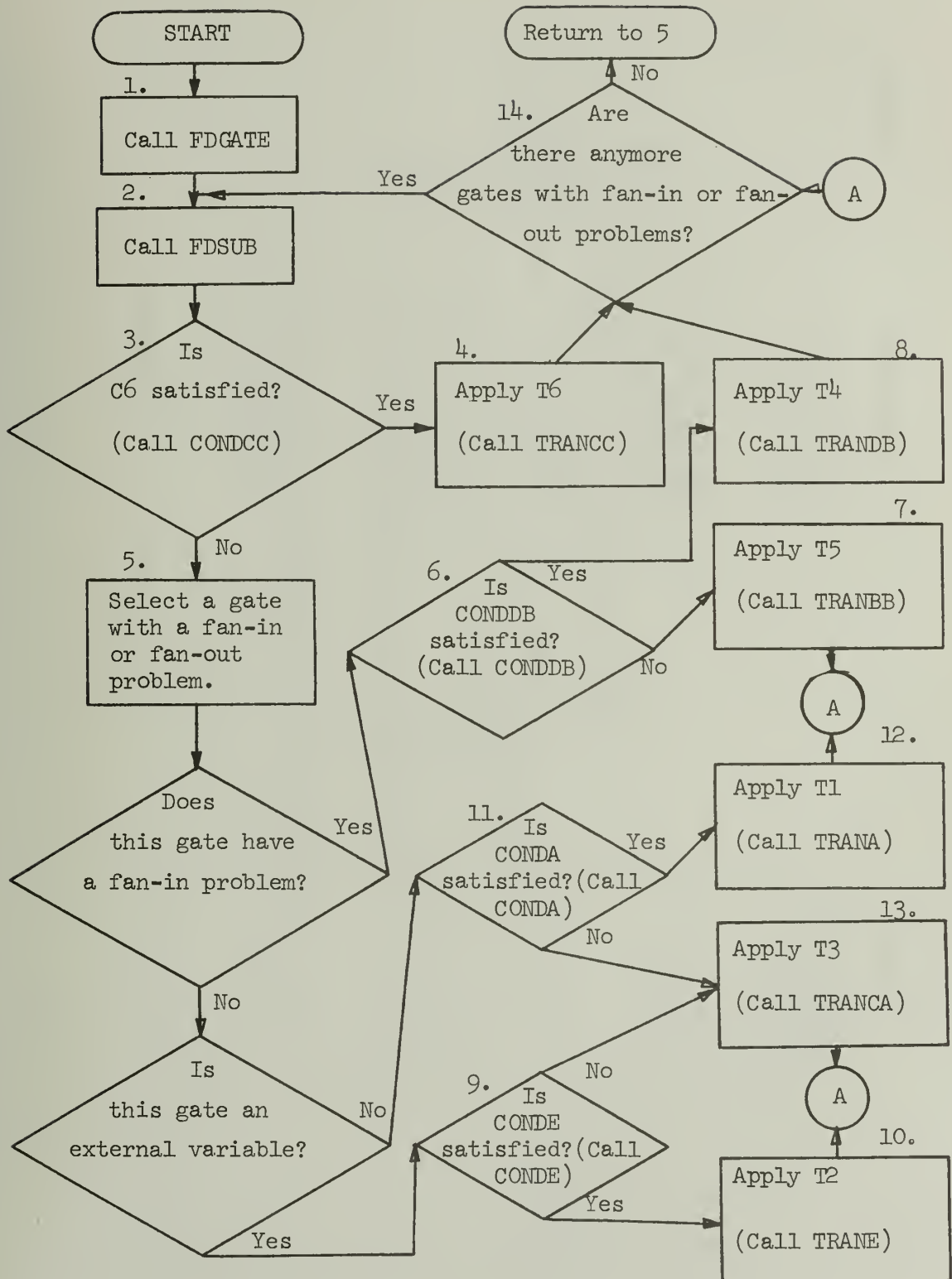


Fig. 3.2.1 Flowchart for subroutine JEFF (Part B).

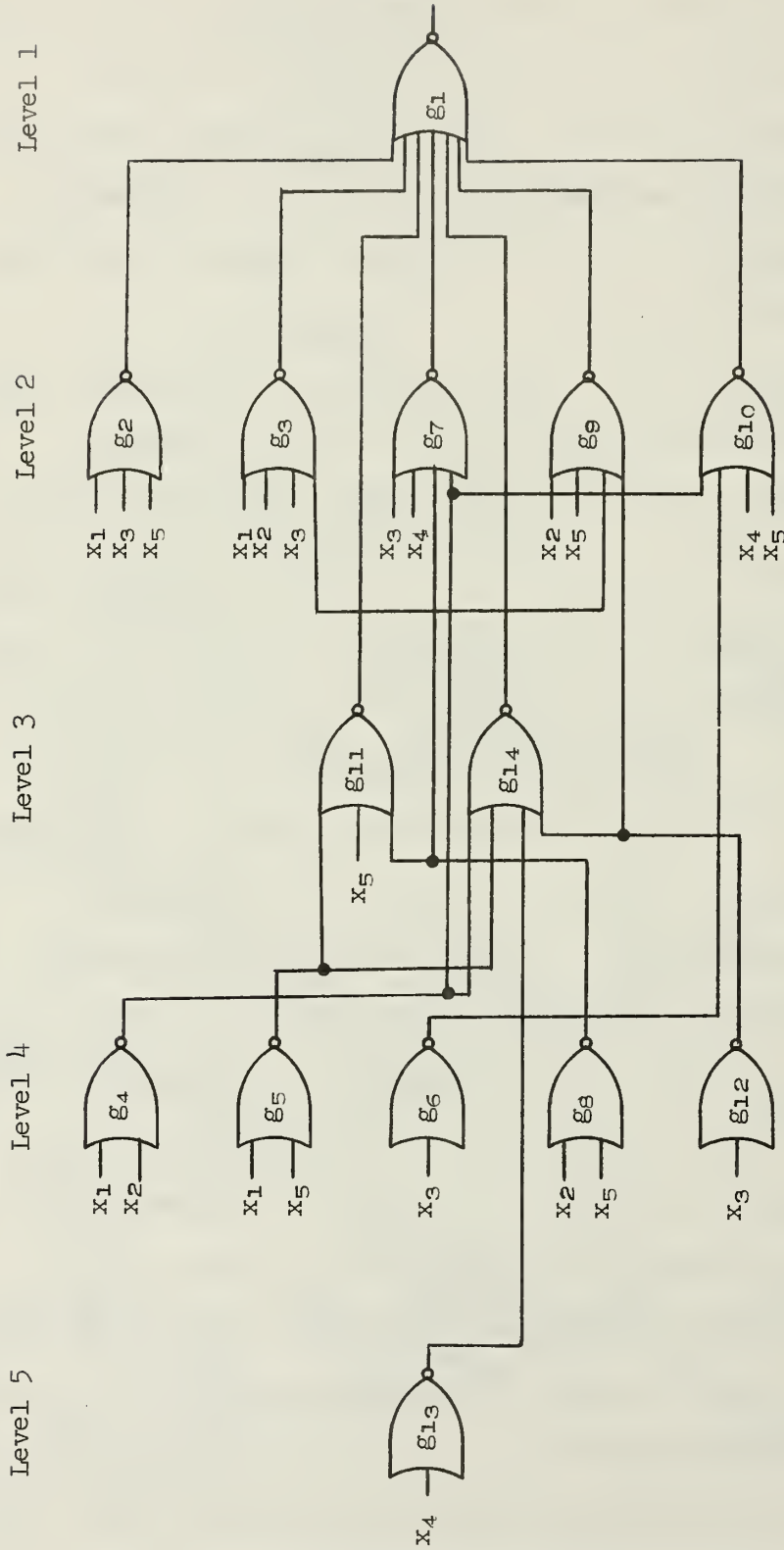


Fig. 3.2.2 The numbering of the levels of a network by PSORT is illustrated.

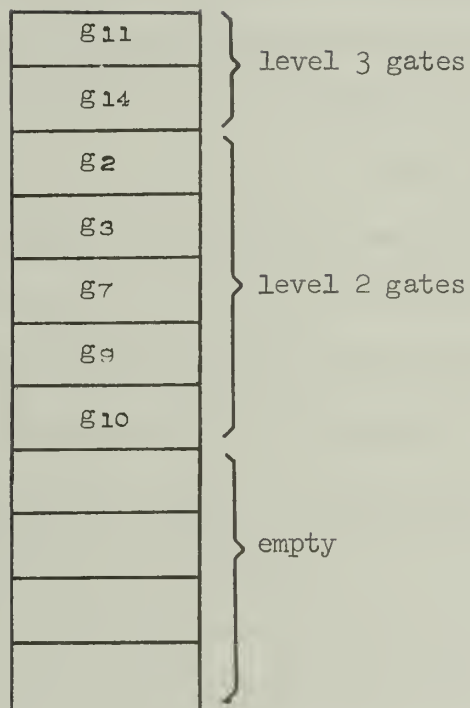


Fig. 3.2.3 The immediate predecessors of g_1 in Fig. 3.2.2 are sorted into descending order according to level number. This gate has 7 immediate predecessors.

the gate.

TRANBB: This subroutine implements transformation T5 which is used to solve the fan-in problem of a specified gate.

TRANCA: This subroutine implements transformation T3 which is used to solve the fan-out problem of a specified gate.

TRANCC: This subroutine implements transformation T6 which is used to solve the composite problem (i.e., it solves fan-in and fan-out problems simultaneously).

TRANDB: This subroutine implements transformation T4 which is used to solve the fan-in problem of a specified gate.

TRANE: This subroutine implements transformation T2 which is used to solve the fan-in problem of a specified gate.

The program which controls Part B of the program FIFOTRAN-G1 is subroutine JEFF. The flowchart for JEFF shown in Fig. 3.2.1 is explained below:

(1) The subroutine FDGATE constructs a list of available gates which can be used for subsequent transformations.

(2) The subroutine FDSUB locates all gates with fan-in and/or fan-out problems.

(3) The subroutine CONDCC searches for gate configurations of the type shown in Fig. 2.3.1.1 and after having found one checks to see if condition set C6 is satisfied. If C6 is not satisfied, then go to (5), otherwise continue to (4).

(4) The subroutine TRANCC applies transformation T6 to the network and the program goes to (14).

(5) The highest level gate with a fan-in or fan-out problem to be resolved is selected next. If this selected gate has a fan-in problem, then go to (6). If the selected gate is really an external variable, then go to (9), otherwise go to (11).

(6) The subroutine CONDDDB checks to see if condition set C4 is satisfied. If yes then go to (8), otherwise continue to (7).

(7) Subroutine TRANBB applies transformation T5 to the selected gate with the fan-in problem.

(8) Subroutine TRANDB applies transformation T4 to the selected gate with the fan-in problem.

(9) The subroutine CONDE checks to see if condition set C2 is satisfied. If yes, then go to (10), otherwise to (13).

(10) The subroutine TRANE applies transformation T2 to the selected gate with the fan-out problem. Go to (14).

(11) The subroutine CONDA checks to see if the condition set C1 is satisfied. If yes, then go to (12), otherwise go to (13).

(12) The subroutine TRANA applies transformation T1 to the selected gate with the fan-out problem. Go to (14).

(13) The subroutine TRANCA applies transformation T3 to the selected gate with the fan-out problem. Go to (14).

(14) If there are any more gates with a fan-in or fan-out problem, then go to (2), otherwise return.

3.3 Flowchart Description of FDSUB and CONDCC

Subroutine FDSUB is responsible for determining which gates have fan-in and/or fan-out problems and deciding in which sequence these problems

will be corrected. Based upon this gate, a subnetwork of connected gates is selected from the existing network, and the number of fan-in and fan-out problems among gates within this subnetwork are determined. This subnetwork is then examined by subroutine CONDCC in order to see if it contains a smaller subnetwork which satisfies condition set C6. If C6 is not satisfied by any smaller subnetwork, then the fan-in or fan-out problem of the selected gate will be solved alone by one of the transformations T_1 , T_2 , T_3 , T_4 , or T_5 .

3.3.1 Subroutine FDSUB

The flowchart for subroutine FDSUB is shown in Fig. 3.3.1.1 and is explained below:

(1) A list of all the gates and external variables with fan-out problems and a list of all the gates with fan-in problems is constructed. The search for gates (or external variables) with fan-in and/or fan-out problems progresses from the highest level to the lowest level (i.e., the output level) of the current network and the gates are listed in the order encountered.

(2) If there are no fan-in or fan-out problems in the given network, then this subroutine returns control to MAIN. Otherwise, it continues to (3).

(3) The level of the first element in the list of gates with fan-in problems is compared with the level of the first element in the list of gates with fan-out problems. If the level of the gate with a fan-out problem is greater than or equal to the level of the gate with a fan-in problem, then the program goes to (5), otherwise it goes to (4). The gate "selected" in this step is designated g_i .

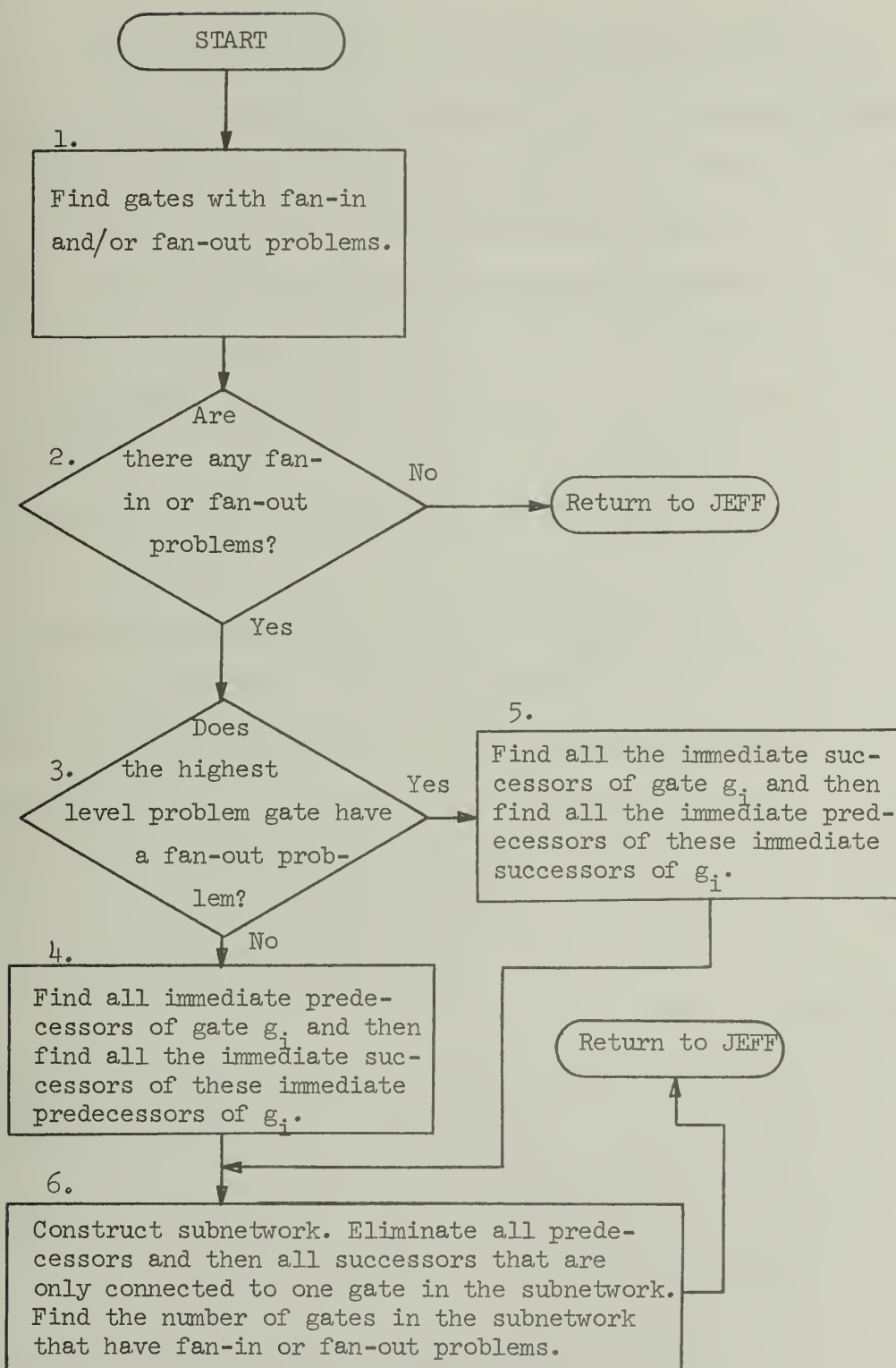


Fig. 3.3.1.1 Flowchart for subroutine FDSUB.

(4) All the immediate predecessors of gate g_i are found. All the immediate successors of these immediate predecessors of g_i are found. These sets of gates will be used in (6). Go to (6).

(5) All the immediate successors of the gate g_i are found. All the immediate predecessors of these immediate successors of g_i are found. These sets of gates will be used in (6). Go to (6).

(6) Using the sets of immediate predecessor and immediate successor gates found in either (4) or (5), eliminate all $IP(g_i)$ (or $IP(IS(g_i))$) with only one immediate successor and then eliminate all $IS(IP(g_i))$ (or $IS(g_i)$) with only one immediate predecessor. From this "reduced" set of immediate predecessor and successor gates, is found the number of gates which have a fan-in problem and the number of gates which have a fan-out problem.

Note: The intersection of the set of immediate predecessor gates and the set of immediate successor gates is empty (i.e., the possibility of having a gate that is both an immediate predecessor and an immediate successor has been precluded by appropriate programming).

3.3.2 Subroutine CONDCC

This subroutine uses the initial subnetwork derived in subroutine FDSUB and searches for the largest contained " $K \times H$ " configuration (See Fig. 2.3.1.1) satisfying condition set C6. The flowchart for subroutine CONDCC is shown in Fig. 3.3.2.1 and is explained below:

(1) A " $K \times H$ " configuration satisfying condition set C6 is found by applying an algorithm (that is not given here) in which criterion for selecting this " $K \times H$ " configuration are included. This is saved.

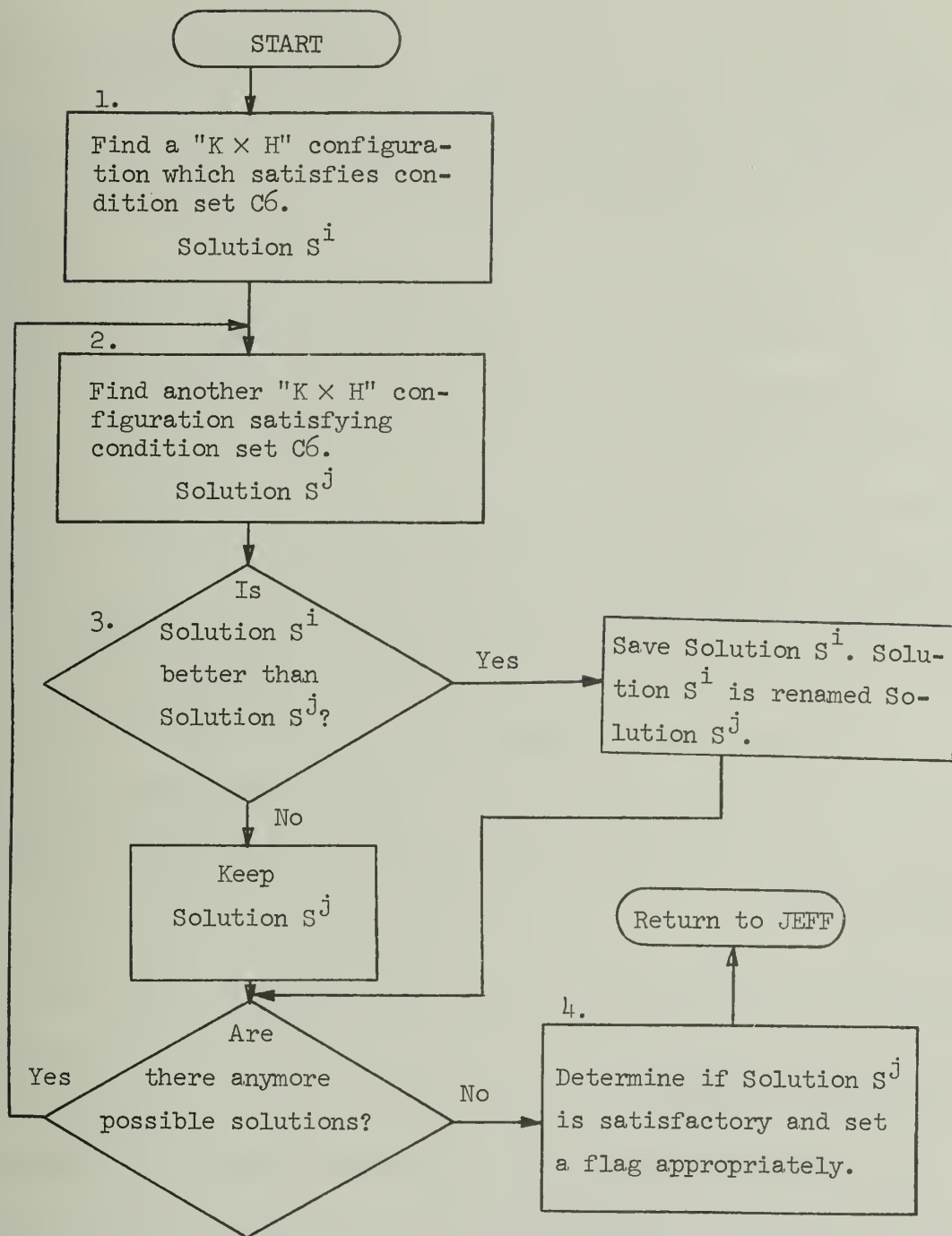


Fig. 3.3.2.1 Flowchart for subroutine CONDCC.

(2) Another " $K \times H$ " configuration satisfying condition set C6 is found by applying the same algorithm referred to in (1).

(3) Compare the number of gates in the " $K \times H$ " configuration found in (2) with the number of gates in the saved " $K \times H$ " configuration. Save the configuration with the largest number of gates. If the configurations have the same number of gates then save the configuration in which the number of "immediate predecessor" gates with a fan-out problem plus the number of "immediate successor" gates with a fan-in problem is greater. If both configurations also have the same number of gates with fan-in and fan-out problems, then keep the configuration that is already "saved". If there still exists a " $K \times H$ " configuration which could have a "solution" that is better than the "saved" solution, then go to (2), otherwise go to (4).

(4) The saved " $K \times H$ " configuration is checked to see if at least one fan-in or fan-out problem will be solved after application of T6. If so, then this " $K \times H$ " configuration satisfies condition set C6. Return control to JEFF.

3.4 How to Use the Program

3.4.1 General Description

Given the program source deck and data deck which will be described in Section 3.4.3, this program reads in a Boolean function or set of functions, finds an initial network, reduces the network, and then solves all the fan-in and fan-out problems that exist in the reduced network by using the transformations described in Section 2 according to the algorithm described in Section 3.1.

3.4.2 Limitations

(1) Networks generated employ all NOR gates (NAND-gate networks can easily be obtained by considering the dual of the desired function to be implemented).

(2) Maximum number of gates plus external variables is 80.

(3) Maximum number of external variables is 5.

(4) "Don't cares" are currently not allowed in the Boolean functions read in by the program.

(5) Both complemented and uncomplemented variables are allowed.

(6) Currently, no restrictions on the number of levels are permitted.

(7) Maximum number of output functions is 80 minus the number of external variables specified.

3.4.3 Source and Data Deck

A typical program deck is shown in Fig. 3.4.3.1. This program deck consists of several JCL cards, a source deck, and a data deck. The source deck includes the main program, Part A subroutines, and Part B subroutines. The data deck is shown in Fig. 3.4.3.2 in more detail and consists of one or more consecutive sets of cards—one set for each network to be synthesized. The set of cards for each network is organized as follows:

(1) Card #1 is the heading card. Any labeling or descriptive information may be given and will be reproduced in the print out. For example,

"**** 5 VAR, EXAMPLE 28"

(2) Card #2 specifies 8 parameters as follows:

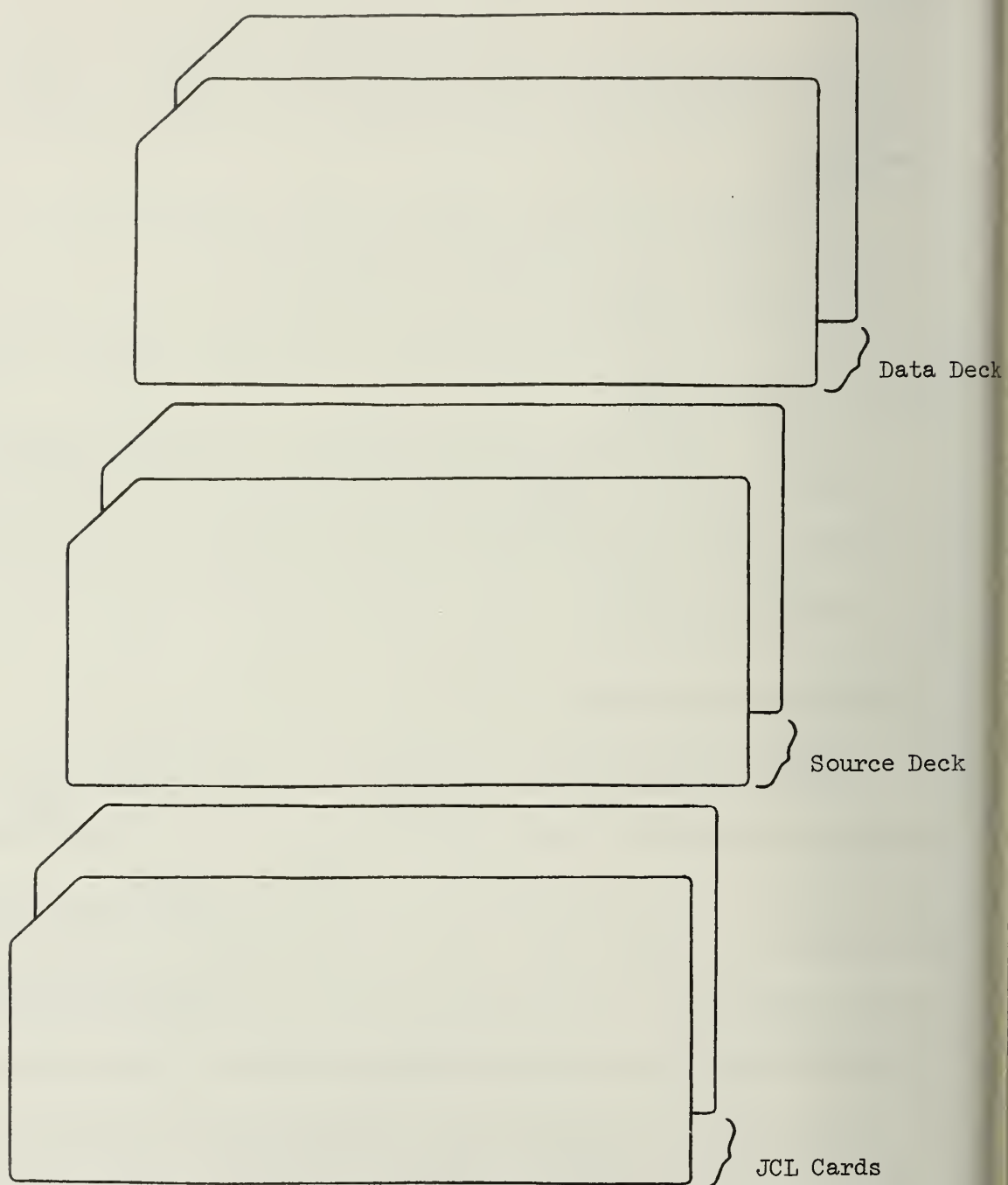


Fig. 3.4.3.1 Typical Program Deck

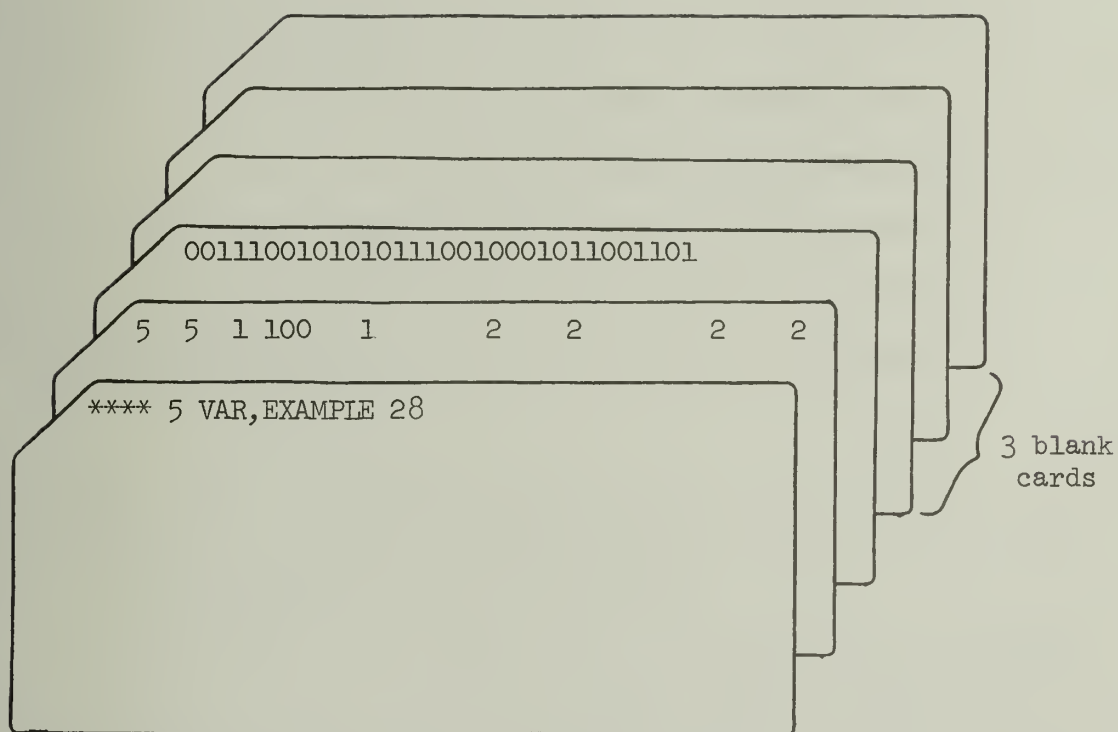


Fig. 3.4.3.2 Sample Data Deck

- (a) Columns 1-4: The number of external variables.
- (b) Columns 5-8: The number of output functions to be realized by the network.
- (c) Columns 9-12: The weight A assigned gates in the network cost function.
- (d) Columns 13-16: The weight B assigned connections in the network cost function.
- (e) Columns 17-20: Any symbol punched here causes the complements of the external variables to be available during the design and transformation procedures.
- (f) Columns 21-24: The fan-in constraint for all gates (FI).
- (g) Columns 25-28: The fan-out constraint for non-output gates (FO).
- (h) Columns 37-40: The fan-out constraint for external variables (FOX).
- (i) Columns 41-44: The fan-out constraint for output gates (FOO).

(3) Card # 3 contains the function to be realized, beginning in column 1. (See Section 4 on how functions are expressed hexadecimally)

(4) Cards #4, #5, and #6 are blank cards.

Groups (3) and (4) must be repeated for each desired output function of a multiple-output network.

4. STATISTICS AND RESULTS

Thirty 4-variable test functions (Group I) and thirty 5-variable test functions (Group II) were selected randomly. Each function in Group I was fed to the program under six different sets of constraints:

- (1) $FI=FO=FOX=FOO=2$
- (2) $FI=FO=FOX=FOO=3$
- (3) $FI=10; FO=FOX=FOO=2$
- (4) $FI=2; FO=FOX=FOO=10$
- (5) $FI=10; FO=FOX=FOO=3$
- (6) $FI=3; FO=FOX=FOO=10$

Similarly, each function in Group II was fed to the program under the following different sets of constraints:

- (1) $FI=FO=FOX=FOO=3$
- (2) $FI=FO=FOX=FOO=4$
- (3) $FI=10; FO=FOX=FOO=3$
- (4) $FI=3; FO=FOX=FOO=10$

In addition to the functions in Group I and Group II a special problem taken from [4] was also tested by the program.

Su and Nam's two-level NAND network which realizes the 4 functions given in Table I of their paper [4] is duplicated here in Fig. 4.1.1. The network realization which they achieved by the use of their fan-in, fan-out problem resolving algorithm is shown in Fig. 4.1.2.

The following alterations were made in order that the current program FIFOTRAN-G1 could be used to solve the same problem:

- (1) Since the current version of FIFOTRAN-G1 is unable to handle incompletely specified functions, several possible combinations of 1's and 0's were substituted for the "don't cares" of Table I of [4]—8 different sets

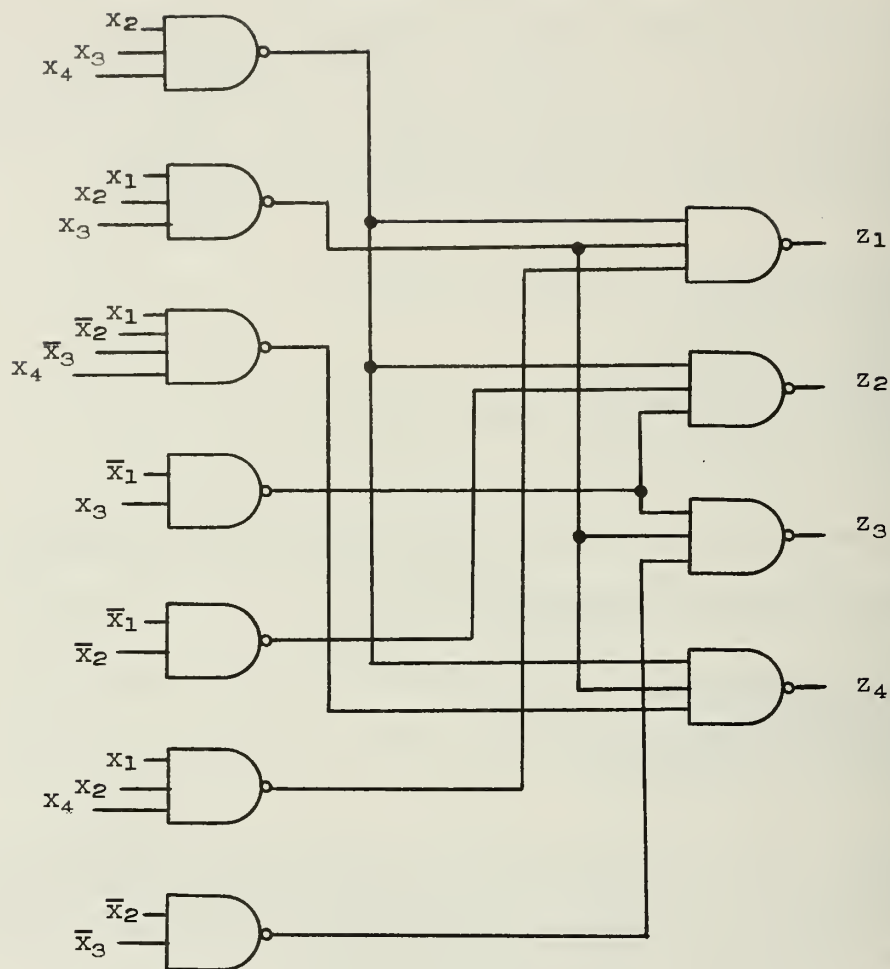


Fig. 4.1.1 Su and Nam's two-level NAND network which realizes the 4 functions given in Table I of their paper [4].

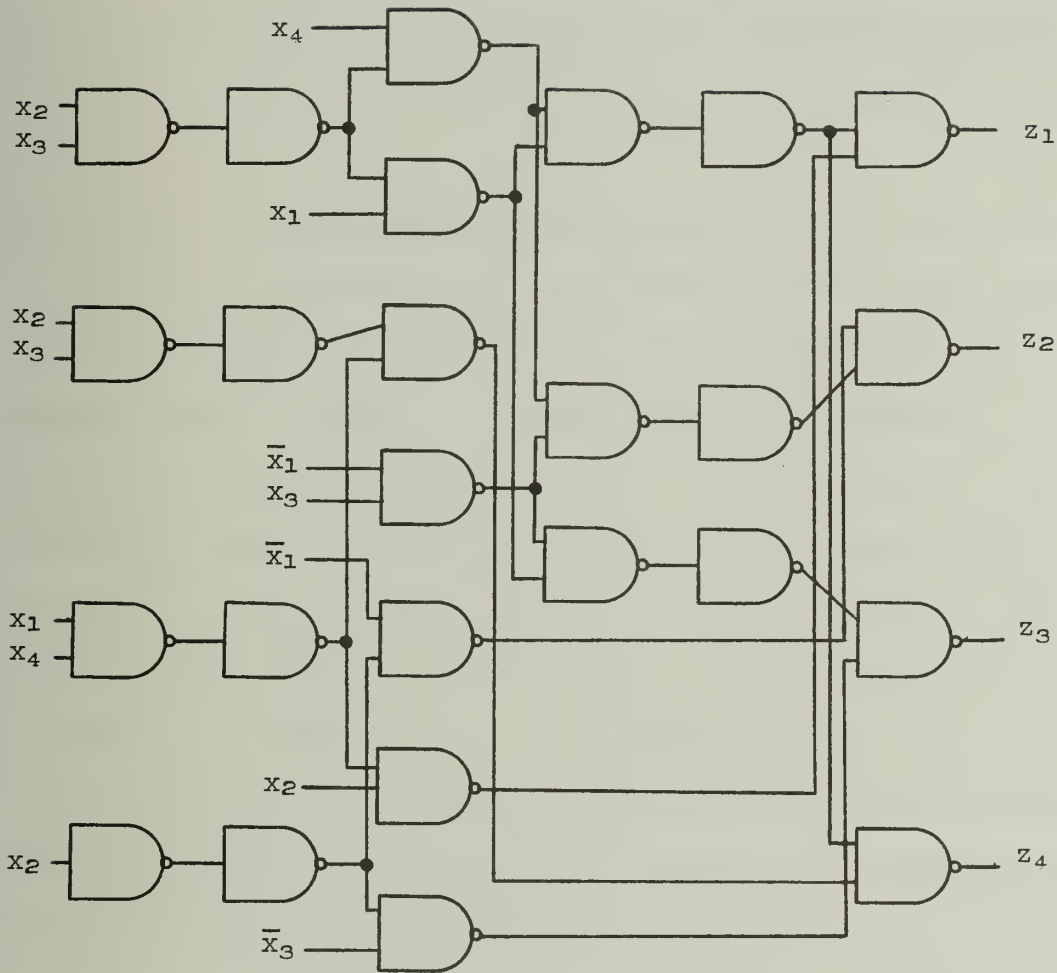


Fig. 4.1.2 Su and Nam's NAND synthesis of the functions in Table I of [4]. (FI = FO = FOX = FOO = 2)

of completely specified functions resulted.

(2) Since FIFOTRAN-G1 employs NOR gates rather than NAND gates, networks were synthesized for the duals of these 8 sets of functions. (See Table 4.1.1)

Each of the 8 different sets of functions in Table 4.1.1 was fed to the program under the same constraints, $FI=FO=FOX=FOO=2$, used in [4].

All test functions are represented here in truth table form using a hexadecimal notation. For example, Fig. 4.1.3 shows the output values of a certain 4-variable function f , corresponding to every possible combination of input values (x_1, x_2, x_3, x_4) . Note that the input combinations considered as binary numbers (i.e., $2^3 \times x_1 + 2^2 \times x_2 + 2^1 \times x_3 + x_4$) start with zero and ascend to 16 in counting order, i.e., 0000, 0001, 0010, ..., 1111. The first four output values (f) that correspond to the first four combinations of input values are used to form the first hexadecimal digit of the hexadecimal representation of the function f as shown in Fig. 4.1.3. Continuing in this manner the resultant hexadecimal representation is found to be 4AF1. (See Fig. 4.1.3)

Table 4.1.2 contains the hexadecimal representations of the test functions in Group I while Table 4.1.3 contains the hexadecimal representations of the functions in Group II.

This section describes the statistics obtained and discusses the results.

4.1 Statistical Analysis

Table 4.1.4 is a table of the statistics compiled for the test functions in Group I with $FI=FO=FOX=FOO=2$. The table consists of 12 columns and

x_1	x_2	x_3	x_4	f	
0	0	0	0	0	} 4
0	0	0	1	1	
0	0	1	0	0	
0	0	1	1	0	
0	1	0	0	1	} A
0	1	0	1	0	
0	1	1	0	1	
0	1	1	1	0	
1	0	0	0	1	} F
1	0	0	1	1	
1	0	1	0	1	
1	0	1	1	1	
1	1	0	0	0	} 1
1	1	0	1	0	
1	1	1	0	0	
1	1	1	1	1	

Fig. 4.1.3 Hexadecimal representation of 4AF1.

#1				#2				#3				#4			
z ₁	z ₂	z ₃	z ₄	z ₁	z ₂	z ₃	z ₄	z ₁	z ₂	z ₃	z ₄	z ₁	z ₂	z ₃	z ₄
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
#5				#6				#7				#8			
z ₁	z ₂	z ₃	z ₄	z ₁	z ₂	z ₃	z ₄	z ₁	z ₂	z ₃	z ₄	z ₁	z ₂	z ₃	z ₄
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
0	1	1	1	0	1	1	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1

Table 4.1.1 Eight dual functions for NOR network synthesis
corresponding to Su and Nam's NAND network synthesis.

1	4	A	F	1
2	F	6	F	E
3	A	C	6	E
4	2	D	8	6
5	9	D	A	5
6	5	F	1	2
7	F	1	F	4
8	6	8	3	0
9	9	0	4	8
10	E	A	9	B
11	6	8	F	5
12	4	6	E	8
13	B	8	6	0
14	E	1	3	9
15	7	0	C	F
16	F	3	D	0
17	B	C	7	A
18	F	6	A	E
19	D	E	8	4
20	6	7	7	7
21	C	1	1	B
22	6	0	C	F
23	D	3	7	9
24	A	E	1	F
25	A	8	4	9
26	B	8	9	6
27	9	B	C	B
28	5	7	1	0
29	6	4	3	3
30	9	9	0	3

Table 4.1.2 Hexadecimal representation of the test functions in Group I.

1	4	F	A	2	9	5	F	6
2	A	6	C	D	D	F	1	8
3	F	F	6	8	A	1	F	3
4	1	E	E	6	5	2	4	0
5	9	E	6	3	B	E	7	F
6	0	A	8	8	8	1	0	3
7	4	9	F	3	6	3	C	D
8	8	B	5	8	0	9	F	0
9	B	F	D	6	C	6	D	A
10	C	6	E	7	1	0	3	E
11	7	A	8	7	1	C	7	1
12	A	E	4	7	B	F	9	F
13	A	B	9	5	6	9	A	2
14	8	8	B	7	4	9	B	4
15	4	9	C	1	3	0	3	1
16	9	6	B	0	3	6	A	5
17	6	8	6	F	2	7	1	F
18	5	B	2	3	D	7	6	3
19	8	5	8	E	C	1	C	B
20	D	5	4	2	D	A	8	6
21	1	1	3	E	5	2	C	2
22	9	6	0	D	6	5	C	8
23	1	D	4	C	0	2	2	D
24	A	6	8	E	9	8	6	6
25	A	8	A	A	5	C	E	8
26	B	4	2	A	9	7	A	8
27	D	5	8	A	9	F	1	1
28	4	D	1	6	B	4	1	4
29	D	A	1	8	2	E	5	1
30	3	9	5	7	2	2	C	D

Table 4.1.3 Hexadecimal representation of the test functions in Group II.

30 rows—one row for each test function. Since there are totally 10 sets of constraints placed on the network synthesis problems for functions in Group I and Group II, there are 10 tables like Table 4.1.4 in the Appendix. The description of each column of Table 4.1.4 is as follows:

Column 1: The number of the test function being used is given here (see Tables 4.1.2 and 4.1.3 to obtain the corresponding hexadecimal representation).

Column 2: A number s in this column indicates that FIFOTRAN-G1 made s iterations of the program loop starting at block 2 in Fig. 3.1.1. During each iteration, FIFOTRAN-G1 takes either the initial network derived by NORNET (if the iteration is the first) or the network derived by JEFF on the previous iteration (otherwise); reduces, without regard to fan-in or fan-out restrictions, the number of gates and connections as far as possible by PROCII; and, finally, uses JEFF to transform the resultant network into one which satisfies the fan-in, fan-out constraints. The algorithm stops looping (i.e., terminates) when the costs of two networks derived by JEFF on consecutive iterations fail to show any improvement (i.e., when the cost of the former is less than or equal to the cost of the latter network). For example, consider function number 5 which had 3 iterations. The cost of the network derived by PROCII and JEFF after each iteration is given in Table 4.1.5. In the first iteration, PROCII reduced the network derived by NORNET and the resultant network cost was 820. JEFF resolved the fan-in, fan-out problems of the reduced network derived by PROCII and the resultant expanded network cost was 2233. In the second iteration, PROCII reduced the previous network derived by JEFF and the resultant network

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	3	46.5	126.0	371	5.0	4.0	821	1930	1109	141.3	135.1
2	2	24.0	123.0	170	2.0	0.0	612	1218	606	99.0	99.0
3	2	54.0	199.0	383	4.0	2.0	920	2334	1414	153.7	153.7
4	3	120.0	421.0	1342	6.5	4.0	1131	3146	2015	206.6	178.2
5	3	49.5	139.0	422	5.0	2.5	820	2131	1311	152.0	159.9
6	2	20.0	130.0	174	2.0	0.0	713	1117	404	56.7	56.7
7	2	32.0	139.0	218	3.0	1.0	714	1421	707	99.0	99.0
8	4	51.7	132.2	638	3.7	3.0	925	1829	904	121.5	97.7
9	3	63.5	174.5	511	4.5	4.0	822	2132	1310	163.8	159.4
10	3	49.5	148.0	383	4.0	1.5	720	1625	905	175.7	125.7
11	2	57.0	122.0	336	5.0	4.0	721	2132	1411	195.7	195.7
12	2	80.0	210.0	614	4.0	4.0	924	3044	2120	229.4	229.4
13	5	48.2	156.5	695	3.5	2.2	822	1321	499	109.0	60.7
14	3	77.0	242.5	787	6.0	4.0	1028	2538	1510	156.5	146.9
15	3	34.0	79.5	225	3.0	1.0	614	1421	807	115.3	131.4
16	2	15.0	140.0	194	2.0	0.0	714	1118	404	56.6	56.6
17	2	55.0	174.0	345	5.0	4.0	822	2436	1614	196.4	196.4
18	3	43.5	110.5	330	4.0	1.5	718	1827	1109	146.0	154.4
19	3	63.5	182.0	522	5.0	2.0	923	2536	1613	175.3	174.8
20	2	14.0	41.0	77	1.0	0.0	510	914	404	79.2	79.2
21	2	49.0	98.0	210	6.0	4.0	720	1930	1210	168.0	168.0
22	2	37.0	94.0	168	5.0	2.0	617	1423	806	130.6	130.6
23	3	88.0	340.5	900	5.0	4.0	927	2233	1306	147.2	140.9
24	3	35.5	91.5	244	4.0	1.5	717	1220	503	119.6	70.2
25	5	72.5	240.5	1193	4.5	2.8	927	2336	1409	140.4	152.0
26	3	88.5	379.0	1058	5.5	4.5	1231	2235	1004	169.8	81.6
27	3	37.0	123.0	328	5.0	2.5	820	1726	906	106.2	110.5
28	2	12.0	117.0	168	1.0	0.0	714	1118	404	56.6	56.6
29	3	28.0	72.5	195	2.0	1.0	614	1219	605	84.7	98.5
30	2	24.0	96.0	148	3.0	1.0	614	1017	403	65.6	65.6
Ave.	2.7	49.0	161.4	445	4.0	2.3				133.9	125.5

Table 4.1.4 Statistics for the test functions in Group I for the case when FI=FO=FOO=FOX=2. These statistics were executed on IBM 360/75J with FORTRAN H (opt 2) compiler.

No. of iterations	Cost of network derived by PROCII	Cost of network derived by JEFF
1	820	2233
2	920	2131
3	920	2131

Table 4.1.5 Network costs derived by PROCII and JEFF after each iteration for function number 5 of Table 4.1.4.

cost was 920. JEFF resolved the fan-in, fan-out problems of this network and the resultant network cost was 2131. In the third iteration, PROCII derived a network (from the previous network derived by JEFF) whose cost was 920. JEFF resolved the fan-in, fan-out problems of this network and the resultant cost was 2131. Since the cost of the networks derived by JEFF on consecutive iterations were equal, the algorithm stops looping. Since the last iteration in the algorithm only causes the termination of the algorithm and is not a part of the sequence leading to the fan-in, fan-out restricted network of least cost, it is excluded from consideration in obtaining most of the statistical results in the table.

Column 3: The average execution time per iteration in subroutine JEFF is given in centiseconds for each function. It is found by dividing the sum of the execution times of JEFF per iteration (excluding the last step) required for that function:

$$t_J = \left[\sum_{i=1}^{s-1} t_{J_i} \right] / (s-1)$$

where t_J denotes the average execution time per iteration in JEFF and t_{J_i} denotes the execution time in centiseconds for JEFF in the i th iteration.

Column 4: The average execution time (t_A) per iteration for each function is given in centiseconds. Let t_{P_i} and t_{J_i} denote the execution times in centiseconds for PROCII and JEFF, respectively, in the i th iteration. Thus for a function requiring totally s iterations, the average execution time per iteration is defined as:

$$t_A = \left[\sum_{i=1}^{s-1} (t_{P_i} + t_{J_i}) \right] / (s-1)$$

Column 5: The total execution time for each function (t_T) is given in centiseconds:

$$t_T = \sum_{i=1}^s (t_{P_i} + t_{J_i})$$

Column 6: For each function, the average number of fan-in problems per iteration is given. In each iteration, JEFF can be thought of as resolving the fan-in and fan-out problems of a "new network" (derived by PROCII), and each "new network" has a certain number of gates with fan-in problems. Summing these numbers, for all except the last iteration, and dividing by the number of iterations minus one gives what is defined as the average number of fan-in problems per iteration.

Column 7: The average number of fan-out problems per iteration is given for each function. It is the sum of the number of fan-out problems per iteration, for all except the last iteration, divided by the number of iterations minus one.

Column 8: The cost of the first network derived by PROCII is given. (See Section 3.1)

Column 9: The cost of the final network derived in the second to the last iteration by JEFF is given. (See Section 3.1)

Column 10: The difference between the final network cost and the original network cost is given for each function—Column 9 minus Column 8.

Column 11: For each function the average increase in network cost per iteration is expressed as a percentage. For the i th iteration ($i = 1, 2, \dots, s$), let C_{P_i} and C_{J_i} be the costs of the network derived by PROCII and JEFF, respectively. Then the average percent increase in cost per iteration expressed in this column is defined as:

$$\left\{ \left[\sum_{i=1}^{s-1} (C_{J_i} - C_{P_i}) / C_{P_i} \right] \times 100 \right\} / (s-1)$$

Column 12: The percentage increase between the original and final network costs is given for each function and is equal to:

$$\frac{C_{J_s - 1} - C_{P_1}}{C_{P_1}} \times 100$$

4.2. Summary of Results

Table 4.2.1 shows the "final averages of the averages" for the test functions in Group I and Group II. Each of the column entries in rows 1-10 of Table 4.2.1 is an average computed from Table 4.1.1. Each of the columns in Table 4.2.1 is averaged and the resultant "final averages of the averages" is given in row 11 of Table 4.2.1. The expression "average average" followed by some quantity indicates that an average was taken over a set of

Test run conditions	Ave. no. of iterations	Ave. ave. time in JEFF per iteration (centiseconds)	Ave. ave. time per iteration (centiseconds)	Ave. total time (centiseconds)	Ave. ave. no. of FAN-IN problems per iteration	Ave. ave. no. of FAN-OUT problems per iteration	Ave. ave. % increase in cost per iteration	Ave. % increase in cost between orig. and final network
4(2,2)	2.7	49.0	161.4	445.0	4.0	2.3	133.9	125.5
4(3,3)	2.3	17.8	109.6	192.5	1.2	0.7	32.4	28.0
4(10,2)	2.6	13.5	102.0	214.5	0.0	2.7	51.3	47.8
4(2,10)	2.4	18.2	114.9	259.0	4.1	0.0	110.8	107.9
4(10,3)	2.2	6.6	93.4	134.4	0.0	0.7	11.7	10.3
4(3,10)	2.1	7.9	98.0	147.1	1.2	0.0	24.6	22.9
5(3,3)	2.9	92.2	788.1	1855.8	5.6	4.1	102.7	94.2
5(4,4)	2.3	41.3	814.8	1190.8	1.6	2.0	37.1	35.0
5(10,3)	2.4	35.2	766.5	1287.6	0.0	4.2	44.4	41.0
5(3,10)	2.5	34.4	741.0	1295.9	5.7	0.0	79.9	71.9
Ave.	2.4	31.6	379.0	702.3	2.3	1.7	62.9	58.4

Table 4.2.1 Summary table containing all the averages of the 10 tables in the Appendix. The expression in column one indicates (column 1, row 1 for example) that the 4-variable test functions in Group I are tested by the program under the constraints FI=2 and FO=F00=FOX=2. The general expression is a(b,c) where a, b, and c are positive integers and a represents the number of variables in the test function, b represents the fan-in constraint, and c represents the fan-out constraint for all gates and external variables.

quantities that were averages. For example, the heading for column 3 of Table 4.2.1 is "Average Average Time In JEFF Per Iteration". This means that the entries in column 3 are averages of the average time in JEFF per iteration. Each of the tables in the Appendix has a row of averages (i.e., row 31). These rows (1 from each of the 10 tables in the Appendix) are simply gathered here in one table.

Table 4.2.2 indicates the number of times a particular transformation was used during a complete run of the test functions in Group I and Group II for a given set of constraints and the total number of all transformations used. For example, when the functions in Group I were run with the constraints FI=FO=FOX=FOO=2, T1 was used 3 times (Col. 1) while the total number of all transformations used was 279.

The entries in Table 4.2.3 indicate the percentage of the number of times a transformation was used during a complete test run. In other words, if a_{ij} represents the i th row and j th column entry of Table 4.2.2, the i th row and j th column entry of Table 4.2.3 (b_{ij}) is:

$$b_{ij} = \frac{a_{ij}}{\sum_{k=1}^6 a_{ik}} \times 100$$

where $i = 1, 2, \dots, 10$ and $j = 1, 2, \dots, 6$.

Test run conditions	T1	T2	T3	T4	T5	T6	Total
4(2,2)	3	15	58	0	154	49	279
4(3,3)	1	6	8	1	34	16	66
4(10,2)	1	19	68	0	0	20	108
4(2,10)	0	0	0	9	127	29	165
4(10,3)	1	11	11	0	0	7	30
4(3,10)	0	1	0	4	32	5	42
5(3,3)	1	43	77	6	135	135	397
5(4,4)	0	11	10	2	44	34	101
5(10,3)	0	45	55	0	0	61	161
5(3,10)	0	0	0	21	110	78	209

Table 4.2.2 This table gives the number of times each transformation (T1-T6) was used during a test run.

Test run conditions	T1	T2	T3	T4	T5	T6
4(2,2)	1.1	5.4	20.8	0.0	55.2	17.6
4(3,3)	1.5	9.1	12.1	1.5	51.5	24.2
4(10,2)	0.9	17.6	63.0	0.0	0.0	18.5
4(2,10)	0.0	0.0	0.0	5.4	77.0	17.6
4(10,3)	3.3	36.7	36.7	0.0	0.0	23.3
4(3,10)	0.0	2.4	0.0	9.5	76.2	11.9
5(3,3)	0.2	10.8	19.4	1.5	34.0	34.0
5(4,4)	0.0	10.9	9.9	2.0	43.6	33.7
5(10,3)	0.0	28.0	34.2	0.0	0.0	37.9
5(3,10)	0.0	0.0	0.0	10.0	52.6	37.3
Average	0.7	12.1	19.6	3.0	39.0	25.6

Table 4.2.3 The percentage of the number of times a transformation was used during a complete test run.

4.3 Su and Nam's Problem

As discussed in the introduction to Section 4, Su and Nam's problem (i.e., the NAND network synthesis of the set of 4 incompletely specified functions in Table I of [4] for the case when $F_I=2$ and $F_O=F_{OO}=F_{OX}=2$) was converted so that it could be treated as a NOR network synthesis problem. Since this program currently can not handle incompletely specified functions, it was necessary to completely specify all functions before FIFOTRAN-G1 could be used. Eight different sets of completely specified functions (See Fig. 4.1.1) were tested, although other completions were also possible. Networks for each of the 8 sets of completely specified functions were synthesized and all fan-in and fan-out problems were resolved.

The best result (i.e., the result which realizes the duals of the 4 functions of Table I of [4] with the fewest number of gates and also satisfies all fan-in and fan-out constraints) is shown in Fig. 4.3.1. This result which is converted into NAND gate form in Fig. 4.3.1 has 19 gates and 33 connections. This shows a 24% improvement over the result obtained by Su and Nam in Fig. 4.1.2: a 25 gate, 33 connection network. Since the external variable $\bar{x}_3$ has only one immediate successor (i.e., $|IP(\bar{x}_3)| = 1$) it is possible to replace gate g_6 by $\bar{x}_3$, thus, an 18 gate, 32 connection network is possible. It should be noted that Su and Nam's network has 6 levels while the network in Fig. 4.3.1 has 10 levels. Also, Su and Nam's output gates are all in level 1 while the output gates in Fig. 4.3.1 are allowed to feed other gates in the network and thus not every output gate is in level 1.

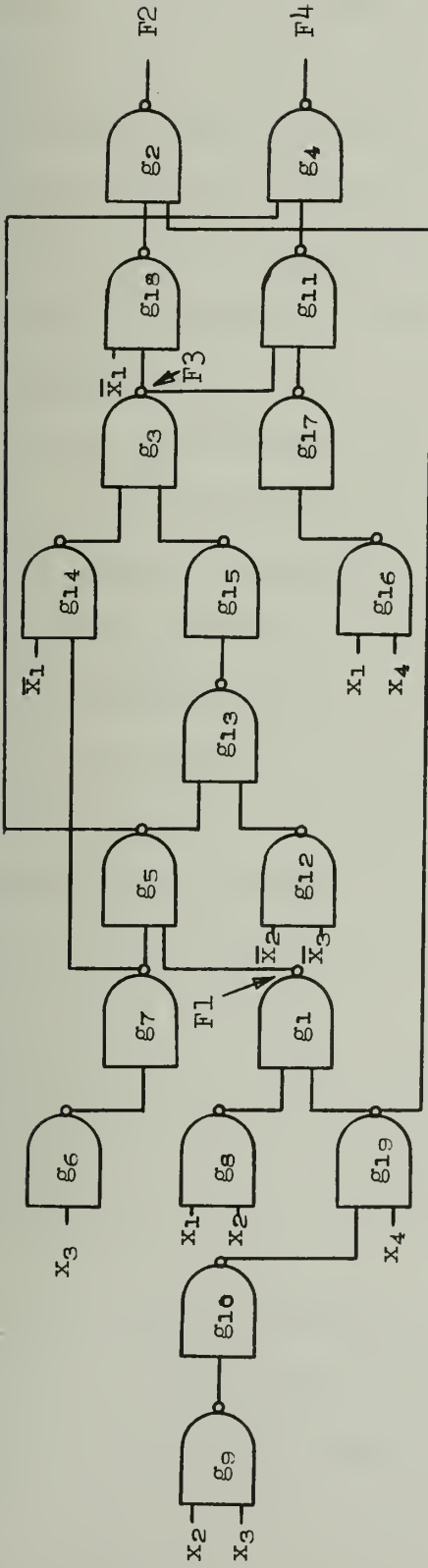


Fig. 4.3.1 A 19 gate, 33 connection solution to Su and Nam's problem (Table I of [4]) when $F1 = F0 = FOX = F00 = 2$ and

$F1 = 0000111101111111$

$F2 = 0111111100110000$

$F3 = 0011110000110000$

$F4 = 0011110101111111$

5. CONCLUSIONS

A computer program (FIFOTRAN-G1) was presented that will synthesize an initial network to realize a given function or set of functions using a simple algorithm, attempt to minimize the number of gates without regard to fan-in and fan-out constraints, and then take the "reduced" network and solve the fan-in and fan-out problems of all gates that exceed the constraints specified by the user. FIFOTRAN-G1's statistics and results for 60 test functions (30 4-variable functions and 30 5-variable functions) run on an IBM 360/75J (all of FIFOTRAN-G1's subroutines were written in FORTRAN IV and compiled by the FORTRAN H (opt 2) compiler) under a variety of fan-in, fan-out constraints were given. In general, the number of gates in a network that realizes a particular function (without regard to fan-in, fan-out constraints) increased approximately 58.4% after using the fan-in, fan-out resolving procedure. (See row 11, column 9 of Table 4.2.1) The greatest average percentage increase (for a group of functions) in the number of gates in a network, 125.5%, occurred when the 4-variable test functions in Group I were tested under the constraints $FI=2$ and $FO=FOO=FOX=2$. (See row 1, column 9 of Table 4.2.1) The smallest average percentage increase (for a group of functions) in the number of gates in a network, 10.3%, occurred when the 4-variable test functions in Group I were tested under the constraints $FI=10$ and $FO=FOO=FOX=3$. (See row 5, column 9 of Table 4.2.1)

Comparing results for the 4-variable functions with those for the 5-variable functions under the same constraints, it can be seen from Table 4.2.1 (column 9) that in general, the percentage increase in cost

between the original network and the final network for the 5-variable functions was about 3 to 4 times greater than that for the 4-variable functions.

Computation times were given for all the test runs. The average total time (for 30 randomly selected test functions) needed to obtain the final fan-in, fan-out constrained network from the initial "reduced" network derived by PROCII was 7.02 seconds. The longest average total time (See row 7, column 5 of Table 4.2.1) 18.55 seconds, occurred when the 5-variable functions were tested under the fan-in, fan-out constraints $FI=3$ and $FO=F00=FOX=3$. The shortest average total time (See row 5, column 5 of Table 4.2.1), 1.34 seconds, occurred when the 4-variable functions were tested under the fan-in, fan-out constraints $FI=10$ and $FO=F00=FOX=3$.

Each average total time (See column 5 of Table 4.2.1) is generally related to the average of the average number of fan-in and fan-out problems per iteration that exist in the network (i.e., column 6 plus column 7). For example, for the two cases just mentioned (i.e., the longest and shortest average total times) the shortest total time corresponds to the "reduced" networks derived by PROCII which have the least average of the average total number of fan-in and fan-out problems (.7) while the longest average total time corresponds to those which have the greatest average of the average total number of fan-in and fan-out problems (9.7). Comparing results for the 4-variable functions with those for the 5-variable functions under the same constraints, it can be seen from column 5 of Table 4.2.1 that in general, the average total time for 5-variable functions was about 8 to 9 times greater than that for 4-variable functions.

From Table 4.2.3 it can be seen that transformation T5 was used in a greater percentage of cases (39%) than any other transformation. Since T1 and T4 (the special transformations, i.e., parallel duplication of a gate and the transformation requiring the existence of a gate whose immediate predecessors are a subset of the immediate predecessors of a gate with a fan-in problem) were used in only .7% and 3.0% of the total number of transformations used, respectively, it can be concluded that these two transformations are not very useful as fan-in, fan-out resolving transformations. These transformations were not used very often since the special conditions which must first be satisfied were not often satisfied. Either these conditions must be relaxed or situations in which these conditions are frequently satisfied must be discovered. However, transformation T2 which solves the fan-out problem of an external variable only was quite useful (representing 12% of the total number of transformations used).

Su and Nam's problem (the problem corresponding to Table I of [4]) was tested by FIFOTRAN-G1 and a network (See fig. 4.3.1) of 19 gates and 33 connections was obtained. FIFOTRAN-G1 includes every type of transformation that Su and Nam's algorithm employs, and it is felt that the fan-in, fan-out resolving transformations are possibly more efficient (i.e., add fewer gates to the network). In addition, FIFOTRAN-G1 has 3 special transformations, of which only one proved to be useful. However, Su and Nam can synthesize larger networks; the current implementation of FIFOTRAN-G1 can not handle functions of more than 5-variables because the transduction procedures can not handle more than 5-variables. The transduction programs can be easily extended such that more variables may be

used. This extension is currently underway. Thus only one of Su and Nam's examples was tested by FIFOTRAN-G1.

APPENDIX

STATISTICS FOR ALL TEST FUNCTIONS IN GROUP I AND GROUP II

In this appendix, tables of statistics like that shown in Table 4.1.4 are given for the test functions in Group I and Group II when run under the 10 sets of constraints given in section 4. There are 10 tables—one for each set of constraints.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	3	46.5	126.0	371	5.0	4.0	821	1930	1109	141.3	135.1
2	2	24.0	123.0	170	2.0	0.0	612	1218	606	99.0	99.0
3	2	54.0	199.0	383	4.0	2.0	920	2334	1414	153.7	153.7
4	3	120.0	421.0	1342	6.5	4.0	1131	3146	2015	206.6	178.2
5	3	49.5	139.0	422	5.0	2.5	820	2131	1311	152.0	159.9
6	2	20.0	130.0	174	2.0	0.0	713	1117	404	56.7	56.7
7	2	32.0	139.0	218	3.0	1.0	714	1421	707	99.0	99.0
8	4	51.7	132.2	638	3.7	3.0	925	1829	904	121.5	97.7
9	3	63.5	174.5	511	4.5	4.0	822	2132	1310	163.8	159.4
10	3	49.5	148.0	383	4.0	1.5	720	1625	905	175.7	125.7
11	2	57.0	122.0	336	5.0	4.0	721	2132	1411	195.7	195.7
12	2	80.0	210.0	614	4.0	4.0	924	3044	2120	229.4	229.4
13	5	48.2	156.5	695	3.5	2.2	822	1321	499	109.0	60.7
14	3	77.0	242.5	787	6.0	4.0	1028	2538	1510	156.5	146.9
15	3	34.0	79.5	225	3.0	1.0	614	1421	807	115.3	131.4
16	2	15.0	140.0	194	2.0	0.0	714	1118	404	56.6	56.6
17	2	55.0	174.0	345	5.0	4.0	822	2436	1614	196.4	196.4
18	3	43.5	110.5	330	4.0	1.5	718	1827	1109	146.0	154.4
19	3	63.5	182.0	522	5.0	2.0	923	2536	1613	175.3	174.8
20	2	14.0	41.0	77	1.0	0.0	510	914	404	79.2	79.2
21	2	49.0	98.0	210	6.0	4.0	720	1930	1210	168.0	168.0
22	2	37.0	94.0	168	5.0	2.0	617	1423	806	130.6	130.6
23	3	88.0	340.5	900	5.0	4.0	927	2233	1306	147.2	140.9
24	3	35.5	91.5	244	4.0	1.5	717	1220	503	119.6	70.2
25	5	72.5	240.2	1193	4.5	2.8	927	2336	1409	140.4	152.0
26	3	88.5	379.0	1058	5.5	4.5	1231	2235	1004	169.8	81.6
27	3	37.0	123.0	328	5.0	2.5	820	1726	906	105.2	110.5
28	2	12.0	117.0	168	1.0	0.0	714	1118	404	56.6	56.6
29	3	28.0	72.5	195	2.0	1.0	614	1219	605	84.7	98.5
30	2	24.0	96.0	148	3.0	1.0	614	1017	403	65.6	65.6
Ave.	2.7	49.0	161.4	445	4.0	2.3				133.9	125.5

Table 1 Statistics for the test functions in Group I for the case when FI=FO=FOO=FOX=2.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave time per iteration (centiseconds)	Total time (centiseconds)	Ave. No. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	14.0	103.0	144	1.0	0.0	821	1023	202	24.6	24.6
2	2	10.0	113.0	146	1.0	0.0	612	814	202	33.0	33.0
3	2	20.0	169.0	225	2.0	0.0	920	1324	404	43.9	43.9
4	4	38.7	152.3	572	3.0	2.7	1131	1532	401	54.8	35.4
5	2	12.0	82.0	123	1.0	0.0	820	1022	202	24.6	24.6
6	2	9.0	121.0	139	0.0	0.0	713	713	0	0.0	0.0
7	2	7.0	112.0	128	0.0	0.0	714	714	0	0.0	0.0
8	3	24.5	125.5	299	1.5	1.0	925	1020	95	43.6	10.3
9	2	15.0	120.0	160	1.0	2.0	822	1023	201	24.4	24.4
10	2	14.0	88.0	136	1.0	0.0	720	922	202	28.0	28.0
11	2	24.0	89.0	153	2.0	1.0	721	1123	402	55.8	55.8
12	2	29.0	154.0	235	1.0	3.0	924	1529	605	65.5	65.5
13	2	22.0	129.0	193	1.0	1.0	822	1226	404	49.1	49.1
14	3	30.5	121.5	327	2.5	1.5	1028	1429	401	59.0	39.0
15	2	9.0	66.0	80	0.0	0.0	614	614	0	0.0	0.0
16	2	9.0	131.0	150	0.0	0.0	714	714	0	0.0	0.0
17	2	18.0	140.0	191	3.0	0.0	822	1225	403	49.0	49.0
18	2	15.0	95.0	133	2.0	0.0	718	919	201	28.0	28.0
19	2	22.0	156.0	220	1.0	2.0	923	1326	403	43.7	43.7
20	2	15.0	44.0	73	1.0	0.0	510	712	203	39.8	39.8
21	3	20.0	68.5	179	1.0	0.5	720	1022	302	40.4	41.9
22	2	9.0	71.0	88	0.0	0.0	617	617	0	0.0	0.0
23	3	36.5	105.0	282	4.0	1.5	927	1528	601	76.2	64.8
24	2	9.0	74.0	98	0.0	0.0	717	717	0	0.0	0.0
25	2	25.0	137.0	219	1.0	3.0	927	1432	505	54.5	54.5
26	4	38.0	165.7	616	3.0	1.7	1231	1631	400	80.8	32.5
27	2	12.0	94.0	135	1.0	0.0	820	1022	202	24.6	24.6
28	2	14.0	116.0	151	1.0	0.0	714	916	202	28.3	28.3
29	2	9.0	68.0	84	0.0	0.0	614	614	0	0.0	0.0
30	2	5.0	77.0	96	0.0	0.0	614	614	0	0.0	0.0
Ave.	2.3	17.8	109.6	192.5	1.2	0.7				32.4	28.0

Table 2 Statistics for the test functions in Group I for the case when FI=FO=FOO=FOX=3.

Function no.	No of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave % increase in cost per iteration	% increase in cost between orig. and final network
1	2	14.0	114.0	199	0.0	4.0	821	1426	605	73.7	73.7
2	2	5.0	125.0	135	0.0	0.0	612	612	0	0.0	0.0
3	2	12.0	166.0	210	0.0	2.0	920	1122	202	22.0	22.0
4	3	32.5	192.5	570	0.0	4.5	1131	2138	1007	90.5	89.0
5	3	15.5	72.5	187	0.0	2.5	820	1223	403	53.4	49.1
6	2	4.0	126.0	137	0.0	0.0	713	713	0	0.0	0.0
7	2	4.0	118.0	134	0.0	1.0	714	815	101	14.1	14.1
8	3	21.0	128.5	373	0.0	4.0	925	1527	602	73.6	65.1
9	2	22.0	142.0	231	0.0	4.0	822	1628	806	98.0	98.0
10	4	8.0	66.3	217	0.0	1.0	720	717	-3	28.1	-.4
11	2	17.0	87.0	166	0.0	4.0	721	1325	604	83.8	83.8
12	2	22.0	169.0	300	0.0	4.0	924	2035	1111	120.2	120.2
13	3	17.0	123.0	323	0.0	4.0	822	1528	706	86.0	85.9
14	4	19.3	125.3	438	0.0	4.0	1028	1326	298	48.2	29.0
15	3	8.0	43.5	117	0.0	1.0	614	815	201	23.5	32.7
16	2	4.0	121.0	137	0.0	4.0	714	714	0	0.0	0.0
17	3	13.0	101.5	250	0.0	3.5	822	1325	503	58.6	61.2
18	3	8.5	55.5	138	0.0	1.5	718	919	201	34.3	28.0
19	3	11.0	101.0	248	0.0	2.0	923	1324	401	54.8	43.4
20	2	4.0	31.0	42	0.0	0.0	510	510	0	0.0	0.0
21	2	15.0	69.0	122	0.0	4.0	720	1324	604	83.9	83.9
22	3	8.5	56.0	152	0.0	1.5	617	1020	403	32.8	65.3
23	2	29.0	107.0	276	0.0	6.0	927	2239	1312	141.5	141.5
24	3	8.0	51.5	134	0.0	1.5	717	817	100	30.4	13.9
25	3	22.5	133.0	360	0.0	5.0	927	1630	433	103.5	46.7
26	2	29.0	164.0	298	0.0	7.0	1231	2340	1109	90.1	90.1
27	3	12.5	72.0	191	0.0	2.5	820	1223	403	53.4	49.1
28	2	4.0	106.0	121	0.0	0.0	714	714	0	0.0	0.0
29	3	7.0	16.5	124	0.0	1.0	614	815	201	23.5	32.7
30	2	7.0	77.0	104	0.0	1.0	614	715	101	16.4	16.4
Ave.	2.6	13.5	102.0	214.5	0.0	2.7				51.3	47.8

Table 3 Statistics for the test functions in Group I for the case when FI=10 and FO=FOO=FOX=2.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	17.0	104.0	171	5.0	0.0	821	1728	907	110.5	110.5
2	2	9.0	111.0	149	2.0	0.0	612	1218	606	99.0	99.0
3	2	20.0	159.0	264	4.0	0.0	920	2132	1212	131.7	131.7
4	3	38.0	249.0	716	6.5	0.0	1131	2540	1409	128.0	124.6
5	2	19.0	88.0	166	5.0	0.0	820	1829	1009	123.0	123.0
6	2	10.0	125.0	156	2.0	0.0	713	1117	404	56.7	56.7
7	2	14.0	121.0	162	3.0	0.0	714	1320	606	84.9	84.9
8	4	17.0	136.3	470	3.7	0.0	925	1526	601	95.2	65.0
9	3	20.0	107.0	271	4.5	0.0	822	1526	704	106.4	85.6
10	3	19.5	93.0	267	4.5	0.0	720	1727	1007	154.6	139.9
11	2	20.0	82.0	177	5.0	0.0	721	1728	1007	139.7	139.7
12	2	22.0	151.0	302	4.0	0.0	924	2034	1110	120.1	120.1
13	3	19.5	111.5	297	4.5	0.0	822	1830	1008	110.6	122.6
14	3	29.5	157.5	479	6.0	0.0	1028	2539	1511	141.3	147.0
15	2	14.0	68.0	111	3.0	0.0	614	1220	606	98.7	98.7
16	2	10.0	125.0	169	2.0	0.0	714	1118	404	56.6	56.6
17	2	20.0	139.0	303	5.0	0.0	822	2235	1413	171.9	171.9
18	2	15.0	82.0	142	4.0	0.0	718	1727	1009	140.5	140.5
19	2	22.0	151.0	263	5.0	0.0	923	2033	1110	120.3	120.3
20	2	9.0	34.0	63	1.0	0.0	510	914	404	79.2	79.2
21	2	19.0	69.0	139	6.0	0.0	720	1728	1008	140.0	140.0
22	2	17.0	84.0	140	5.0	0.0	617	1221	604	97.9	97.9
23	2	35.0	112.0	341	6.0	0.0	927	2743	1816	195.9	195.9
24	3	14.5	75.0	200	4.0	0.0	717	1220	503	105.6	70.2
25	4	23.0	160.7	604	5.3	0.0	927	2133	1206	108.2	130.1
26	3	32.0	222.5	674	6.0	0.0	1231	2439	1208	131.3	98.1
27	2	14.0	93.0	157	5.0	0.0	820	1626	806	98.3	98.3
28	2	7.0	111.0	143	1.0	0.0	714	1118	404	56.6	56.6
29	2	10.0	64.0	102	2.0	0.0	614	1018	404	65.8	65.8
30	3	10.0	62.0	173	2.5	0.0	614	1016	402	54.0	65.5
Ave.	2.4	18.2	114.9	259.0	4.1	0.0				110.8	107.9

Table 4 Statistics for the test functions in Group I for the case when FI=2 and FO=FOO=FOX=10.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)		Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	4.0	93.0	111	0.0	0.0	821	821	0	0.0	0.0	0.0
2	2	4.0	108.0	123	0.0	0.0	612	612	0	0.0	0.0	0.0
3	2	4.0	148.0	164	0.0	0.0	920	920	0	0.0	0.0	0.0
4	5	14.2	84.0	382	0.0	2.8	1131	1229	98	27.7	8.7	8.7
5	2	5.0	75.0	93	0.0	0.0	820	820	0	0.0	0.0	0.0
6	2	4.0	118.0	130	0.0	0.0	713	713	0	0.0	0.0	0.0
7	2	7.0	112.0	125	0.0	0.0	714	714	0	0.0	0.0	0.0
8	2	12.0	127.0	141	0.0	2.0	925	1228	303	32.8	32.8	32.8
9	2	12.0	119.0	170	0.0	2.0	822	1226	404	49.1	49.1	49.1
10	2	4.0	79.0	95	0.0	0.0	720	720	0	0.0	0.0	0.0
11	2	9.0	73.0	104	0.0	1.0	721	923	202	28.0	28.0	28.0
12	2	12.0	137.0	188	0.0	3.0	924	1327	403	43.6	43.6	43.6
13	2	7.0	114.0	163	0.0	1.0	822	1024	202	24.6	24.6	24.6
14	3	6.5	82.0	182	0.0	1.0	1028	1025	-3	9.8	-.3	-.3
15	2	4.0	59.0	71	0.0	0.0	614	614	0	0.0	0.0	0.0
16	2	4.0	119.0	134	0.0	0.0	714	714	0	0.0	0.0	0.0
17	2	4.0	121.0	135	0.0	0.0	822	822	0	0.0	0.0	0.0
18	2	5.0	74.0	85	0.0	0.0	718	718	0	0.0	0.0	0.0
19	2	7.0	136.0	183	0.0	2.0	923	1124	201	21.8	21.8	21.8
20	2	5.0	29.0	40	0.0	0.0	510	510	0	0.0	0.0	0.0
21	2	9.0	59.0	88	0.0	1.0	720	922	202	28.0	28.0	28.0
22	2	4.0	63.0	74	0.0	0.0	617	617	0	0.0	0.0	0.0
23	3	12.0	71.0	181	0.0	2.0	927	1229	302	31.6	32.6	32.6
24	2	4.0	63.0	81	0.0	0.0	717	717	0	0.0	0.0	0.0
25	2	10.0	117.0	171	0.0	3.0	927	1230	303	32.7	32.7	32.7
26	3	9.5	99.0	240	0.0	1.5	1231	1332	101	21.2	8.2	8.2
27	2	4.0	83.0	99	0.0	0.0	820	820	0	0.0	0.0	0.0
28	2	4.0	108.0	121	0.0	0.0	714	714	0	0.0	0.0	0.0
29	2	4.0	59.0	72	0.0	0.0	614	614	0	0.0	0.0	0.0
30	2	4.0	74.0	87	0.0	0.0	614	614	0	0.0	0.0	0.0
Ave.	2.2	6.6	93.4	134.4	0.0	0.7					11.7	10.3

Table 5 Statistics for the test functions in Group I for the case when FI=10 and FO=F00=FOX=3.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	7.0	96.0	132	1.0	0.0	821	1023	202	24.6	24.6
2	2	7.0	112.0	136	1.0	0.0	612	814	202	33.0	33.0
3	2	10.0	147.0	191	2.0	0.0	920	1324	404	43.9	43.9
4	3	12.5	117.5	295	3.0	0.0	1131	1633	502	43.2	44.4
5	2	7.0	77.0	111	1.0	0.0	820	1022	202	24.6	24.6
6	2	3.0	113.0	126	0.0	0.0	713	713	0	0.0	0.0
7	2	4.0	113.0	126	0.0	0.0	714	714	0	0.0	0.0
8	3	8.5	105.5	254	1.5	0.0	925	1020	95	27.3	10.3
9	2	7.0	114.0	146	1.0	0.0	822	922	100	12.2	12.2
10	2	7.0	80.0	117	1.0	0.0	720	922	202	28.0	28.0
11	2	10.0	72.0	117	2.0	0.0	721	1022	301	41.7	41.7
12	2	7.0	131.0	175	1.0	0.0	924	1126	202	21.9	21.9
13	2	7.0	112.0	156	1.0	0.0	822	1225	403	49.0	49.0
14	3	12.5	92.0	249	2.5	0.0	1028	1429	401	49.2	39.0
15	2	4.0	59.0	70	0.0	0.0	614	614	0	0.0	0.0
16	2	5.0	122.0	138	0.0	0.0	714	714	0	0.0	0.0
17	2	10.0	132.0	171	3.0	0.0	822	1225	403	49.0	49.0
18	2	9.0	81.0	112	2.0	0.0	718	919	201	28.0	28.0
19	2	10.0	142.0	185	1.0	0.0	923	1125	202	21.9	21.9
20	2	7.0	40.0	66	1.0	0.0	510	712	202	39.6	39.6
21	2	7.0	59.0	87	1.0	0.0	720	922	202	21.9	21.9
22	2	5.0	65.0	78	0.0	0.0	617	617	0	0.0	0.0
23	2	17.0	104.0	170	4.0	0.0	927	1430	503	54.3	54.3
24	2	4.0	63.0	82	0.0	0.0	717	717	0	0.0	0.0
25	2	7.0	116.0	159	1.0	0.0	927	1129	202	21.8	21.8
26	3	15.0	135.5	338	3.0	0.0	1231	1531	300	50.6	24.4
27	2	7.0	86.0	120	1.0	0.0	820	1022	202	24.6	24.6
28	2	9.0	114.0	141	1.0	0.0	714	916	202	28.3	28.3
29	2	4.0	64.0	76	0.0	0.0	614	614	0	0.0	0.0
30	2	7.0	77.0	90	0.0	0.0	614	614	0	0.0	0.0
Ave.	2.1	7.9	98.0	147.1	1.2	0.0				24.6	22.9

Table 6 Statistics for the test functions in Group I for the case when FI=3 and FO=F00=FOX=10.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	109.0	1274.0	1712	7.0	6.0	1443	3158	1715	118.8	118.8
2	2	159.0	1722.0	2495	8.0	6.0	1648	3967	2319	140.7	140.7
3	2	84.0	793.0	1142	6.0	5.0	1036	2041	1005	97.0	97.0
4	3	88.0	745.5	1658	3.0	4.0	1234	2038	804	98.2	65.2
5	3	87.0	612.0	1572	7.0	3.5	1237	2445	1208	114.0	97.6
6	3	40.0	414.0	932	2.5	1.5	931	1330	399	47.2	42.8
7	2	89.0	797.0	1179	8.0	7.0	1342	2854	1512	112.7	112.7
8	3	79.5	678.0	1689	4.0	3.0	1235	2141	906	77.6	73.3
9	3	106.0	851.5	2139	7.0	4.5	1444	2850	1406	115.6	97.4
10	2	95.0	103.9	1453	6.0	5.0	1337	2748	1411	105.5	105.5
11	3	128.5	769.5	2232	9.0	6.0	1550	3462	1912	130.4	123.4
12	2	70.0	412.0	709	7.0	5.0	1036	2243	1207	116.5	116.5
13	3	123.0	980.5	2379	8.0	6.0	1548	3565	2017	117.0	130.3
14	5	110.2	797.0	3483	6.2	3.0	1747	2240	493	102.5	28.2
15	2	37.0	748.0	919	2.0	2.0	1029	1432	403	39.2	39.2
16	3	89.5	660.0	1809	6.0	3.0	1344	3057	1713	116.6	127.4
17	5	109.8	518.5	2659	6.0	6.0	1245	2750	1505	153.0	120.9
18	2	67.0	602.0	886	6.0	3.0	1237	2244	1007	81.4	81.4
19	3	44.5	518.5	1157	3.0	0.5	1129	1732	603	62.7	53.4
20	3	101.0	868.0	2043	7.0	6.5	1545	3157	1612	107.6	104.3
21	3	76.0	771.0	1938	2.5	3.0	1437	2445	1008	82.4	70.1
22	2	100.0	1210.0	1713	8.0	5.0	1647	3461	1814	110.1	110.1
23	3	60.0	823.5	1942	2.0	4.0	1543	2246	703	70.6	45.6
24	2	129.0	1216.0	1757	6.0	3.0	1339	3053	1714	128.0	128.0
25	3	51.0	755.5	1653	3.0	2.5	1131	1834	703	69.2	62.2
26	3	145.0	1199.5	3242	7.0	4.0	1548	3664	2116	124.1	136.7
27	3	64.0	493.5	1241	4.0	2.0	1136	2039	903	88.6	79.5
28	5	87.8	563.2	2533	4.8	3.0	1442	2140	698	92.3	48.4
29	2	134.0	953.0	1739	7.0	7.0	1346	3565	2219	164.8	164.8
30	5	103.2	793.0	3668	4.8	4.2	1546	3160	1614	96.2	104.4
Ave.	2.9	92.2	788.1	1855.8	5.6	4.1				102.7	94.2

Table 7 Statistics for the test functions in Group II for the case when FI=FO=FOO=FOX=3.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	49.0	1038.0	1240	2.0	2.0	1443	1947	504	34.9	34.9
2	2	34.0	1081.0	1328	2.0	2.0	1648	2252	604	36.6	36.6
3	2	37.0	713.0	875	3.0	1.0	1036	1439	403	38.9	38.9
4	2	45.0	1019.0	1227	1.0	3.0	1234	1838	604	48.9	48.9
5	3	40.5	363.0	850	3.0	1.5	1237	1538	301	42.1	24.3
6	2	20.0	659.0	742	1.0	0.0	931	1133	202	21.7	21.7
7	2	37.0	738.0	891	1.0	2.0	1342	1744	402	30.0	30.0
8	3	24.0	570.5	1300	1.5	0.0	1235	1536	301	23.9	24.4
9	3	56.0	704.5	1662	4.5	2.5	1444	1844	400	50.2	27.7
10	2	32.0	969.0	1091	1.0	1.0	1337	1640	303	22.7	22.7
11	3	57.5	653.0	1564	1.0	3.5	1550	2353	803	58.5	51.8
12	3	25.5	289.5	680	1.0	0.5	1036	1336	300	28.4	29.0
13	2	49.0	1010.0	1252	3.0	3.0	1548	2352	804	51.9	51.9
14	2	49.0	1015.0	1334	1.0	4.0	1747	2553	806	46.1	46.1
15	2	20.0	724.0	809	1.0	0.0	1029	1231	202	19.6	19.6
16	3	50.5	548.0	1400	2.0	3.0	1344	1946	602	39.8	44.8
17	2	70.0	388.0	910	2.0	4.0	1245	2253	1008	81.0	81.0
18	2	37.0	583.0	729	1.0	2.0	1237	1639	402	32.5	32.5
19	2	22.0	778.0	876	1.0	0.0	1129	1331	202	17.9	17.9
20	2	39.0	1287.0	1535	1.0	2.0	1545	2149	604	39.1	39.1
21	2	35.0	1279.0	1406	1.0	1.0	1437	1740	303	21.1	21.1
22	2	60.0	1217.0	1426	2.0	4.0	1647	2350	703	42.7	42.7
23	2	40.0	1001.0	1192	1.0	1.0	1543	2049	506	32.8	32.8
24	2	57.0	1125.0	1351	2.0	3.0	1339	1943	604	45.1	45.1
25	2	20.0	1315.0	1468	1.0	0.0	1131	1333	202	17.9	17.9
26	2	54.0	1174.0	1406	2.0	4.0	1548	2150	602	38.9	38.9
27	3	33.0	446.0	1011	1.0	1.0	1136	1537	401	30.1	35.3
28	3	39.5	576.5	1329	1.0	1.5	1442	1946	504	26.7	35.0
29	3	52.5	627.0	1603	1.5	2.5	1346	2050	704	46.4	52.3
30	3	53.0	552.5	1238	2.0	2.5	1546	1639	93	45.6	6.0
Ave.	2.3	41.3	814.8	1190.8	1.6	2.0				37.1	35.0

Table 8 Statistics for the test functions in Group II for the case when FI=FO=F00=FOX=4.

Function no.	No. of iterations	Ave time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	42.0	1025.0	1287	0.0	6.0	1443	2146	703	48.7	48.7
2	3	49.5	771.0	1919	0.0	5.5	1648	2555	907	61.1	55.0
3	2	30.0	704.0	949	0.0	5.0	1036	1639	603	58.2	58.2
4	3	34.5	785.0	1702	0.0	4.0	1234	1636	402	57.2	32.6
5	2	32.0	658.0	793	0.0	3.0	1237	1641	404	32.7	32.7
6	3	18.0	423.5	883	0.0	1.5	931	1028	97	21.6	10.4
7	2	35.0	836.0	1024	0.0	7.0	1342	2045	703	52.4	52.4
8	2	42.0	1104.0	1309	0.0	4.0	1235	1841	606	49.1	49.1
9	3	34.5	682.0	1542	0.0	3.0	1444	1842	398	43.0	27.6
10	2	39.0	1094.0	1276	0.0	5.0	1337	1940	603	45.1	45.1
11	2	74.0	890.0	1436	0.0	7.0	1550	2860	1310	84.5	84.5
12	2	29.0	397.0	611	0.0	5.0	1036	1637	601	58.0	58.0
13	2	40.0	1090.0	1504	0.0	6.0	1548	2250	702	45.3	45.3
14	2	40.0	1074.0	1295	0.0	4.0	1747	2252	505	28.9	28.9
15	3	18.5	358.5	818	0.0	2.0	1029	1229	200	19.2	19.4
16	3	28.5	538.5	1319	0.0	3.0	1344	1946	602	44.1	44.8
17	5	38.5	300.2	1451	0.0	4.5	1245	1839	594	60.6	47.7
18	3	23.0	333.0	785	0.0	3.0	1237	1638	401	27.5	32.4
19	3	14.5	416.0	926	0.0	1.0	1129	1330	201	13.0	17.8
20	3	40.5	777.5	1803	0.0	5.5	1545	2148	603	50.4	39.0
21	2	34.0	1440.0	1582	0.0	4.0	1437	1942	505	35.1	35.1
22	2	49.0	1153.0	1369	0.0	5.0	1647	2352	705	42.8	42.8
23	3	26.0	695.5	1591	0.0	4.0	1543	1638	95	29.8	6.2
24	2	28.0	1044.0	1223	0.0	3.0	1339	1743	404	30.2	30.2
25	3	18.0	674.5	1447	0.0	2.5	1131	1430	299	26.0	26.4
26	3	47.0	773.0	1898	0.0	5.0	1548	2555	1007	55.0	65.1
27	3	22.0	400.0	901	0.0	2.5	1136	1436	300	34.8	26.4
28	2	34.0	1015.0	1262	0.0	4.0	1442	2147	705	48.9	48.9
29	2	47.0	881.0	1189	0.0	7.0	1346	2352	1006	74.7	74.7
30	3	48.0	660.5	1535	0.0	5.5	1546	2252	706	54.2	45.7
Ave.	2.4	35.2	766.5	1287.6	0.0	4.2				44.4	41.0

Table 9 Statistics for the test functions in Group II for the case when FI=10 and FO=F00=FOX=3.

Function no.	No. of iterations	Ave. time in JEFF per iteration (centiseconds)	Ave. time per iteration (centiseconds)	Total time (centiseconds)	Ave. no. of FAN-IN problems per iteration	Ave. no. of FAN-OUT problems per iteration	Orig. network cost	Final network cost	Cost difference	Ave. % increase in cost per iteration	% increase in cost between orig. and final network
1	2	37.0	1004.0	1291	7.0	0.0	1443	2856	1413	97.9	97.9
2	2	47.0	1099.0	1506	8.0	0.0	1648	3261	1613	97.9	97.9
3	2	30.0	693.0	932	6.0	0.0	1036	1737	701	67.7	67.7
4	2	30.0	1014.0	1156	4.0	0.0	1234	1940	706	57.2	57.2
5	3	49.5	438.5	1180	7.0	0.0	1237	2750	1513	136.3	122.3
6	3	19.0	403.0	897	2.5	0.0	931	1330	399	36.3	42.8
7	2	34.0	742.0	1016	8.0	0.0	1342	2551	1209	90.1	90.1
8	3	32.0	609.0	1355	4.0	0.0	1235	1939	704	61.3	57.0
9	4	39.0	538.3	1786	6.3	0.0	1444	2143	699	77.9	48.4
10	3	33.5	652.0	1503	6.0	0.0	1337	2142	805	78.1	60.2
11	2	50.0	769.0	1266	10.0	0.0	1550	3061	1511	97.5	97.5
12	2	30.0	365.0	644	7.0	0.0	1036	2042	1006	97.1	97.1
13	2	44.0	1009.0	1326	8.0	0.0	1548	3160	1612	104.1	104.1
14	5	41.5	677.2	2947	7.0	0.0	1747	2651	904	86.0	51.7
15	2	19.0	722.0	821	2.0	0.0	1029	1330	301	29.2	29.2
16	2	45.0	843.0	1134	6.0	0.0	1344	2757	1413	105.1	105.1
17	3	38.5	325.5	882	6.0	0.0	1245	2450	1205	98.2	96.8
18	2	30.0	531.0	776	6.0	0.0	1237	2043	806	65.2	65.2
19	2	24.0	760.0	894	3.0	0.0	1129	1735	606	53.7	53.7
20	3	36.5	736.0	1714	7.5	0.0	1545	2855	1310	82.3	84.8
21	3	23.5	687.0	1609	2.5	0.0	1437	2142	705	54.9	49.1
22	2	42.0	1146.0	1583	8.0	0.0	1647	3262	1615	98.0	98.0
23	3	20.0	582.0	1381	2.0	0.0	1543	2148	605	50.8	39.2
24	2	39.0	1060.0	1332	6.0	0.0	1339	2547	1208	47.4	47.4
25	2	25.0	1234.0	1362	3.0	0.0	1131	1837	706	62.4	62.4
26	3	38.5	708.0	1730	6.5	0.0	1548	2346	798	77.0	51.5
27	3	27.0	418.5	979	4.0	0.0	1136	1838	702	64.4	61.8
28	2	32.0	971.0	1245	6.0	0.0	1442	2552	1110	77.0	77.0
29	2	40.0	872.0	1172	7.0	0.0	1346	2759	1413	105.0	105.0
30	3	34.5	619.5	1458	5.5	0.0	1546	2142	596	82.2	38.6
Ave.	2.5	34.4	741.0	1295.9	5.7	0.0				79.9	71.9

Table 10 Statistics for the test functions in Group II for the case when FI=3 and FO=F00=FOX=10.

LIST OF REFERENCES

- [1] D.L. Dietmeyer and P.R. Schneider, "A computer-oriented factoring algorithm for NOR logic design," IEEE Trans. Electron. Comput., Vol. EC-14, pp. 868-874, Dec. 1965.
- [2] D.L. Dietmeyer and Y.H. Su, "Logic design automation of fan-in limited NAND networks," IEEE Trans. Comput., Vol. C-18, pp. 11-22, Jan. 1969.
- [3] Y.H. Su, "Multiple-output NAND networks," Electronic Design 23, pp. 98-105, Nov. 8, 1973.
- [4] Y.H. Su and C.W. Nam, "Computer-aided synthesis of multiple-output multi-level NAND networks with fan-in and fan-out constraints," IEEE Trans. Comput., Vol. C-20, pp. 1445-1455, Dec. 1971.

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-74-661	2.	3. Recipient's Accession No.
	Title and Subtitle THE DESIGN OF NOR NETWORKS UNDER FAN-IN AND FAN-OUT CONSTRAINTS (PROGRAMMING MANUAL FOR FIFOTRAN-G1)		5. Report Date June, 1974
Author(s) Jeffry Garnet Legge		6.	8. Performing Organization Rept. No.
Performing Organization Name and Address Department of Computer Science University of Illinois at Champaign-Urbana Urbana, Illinois 61801		10. Project/Task/Work Unit No.	11. Contract/Grant No. NSF Grant No. NSF-GJ-40221
2. Sponsoring Organization Name and Address National Science Foundation 1800 G. Street, N.W. Washington, D.C. 20550		13. Type of Report & Period Covered Technical Report	14.
i. Supplementary Notes			
j. Abstracts This report presents a simple algorithm (contained in the program FIFOTRAN-G1) which can find minimal NOR (NAND) networks while taking into account the fan-in and fan-out constraints of a NOR (NAND) gate. The capability of designing NOR networks under fan-in and fan-out constraints is combined with transduction programs (written by S. Muroga's research group). FIFOTRAN-G1 can convert any multiple-output, multi-level NOR network that does not satisfy specified fan-in, fan-out constraints into a network which satisfies the constraints by applying appropriate transformations. This report serves as a programming manual for FIFOTRAN-G1.			
k. Key Words and Document Analysis. 17a. Descriptors fan-in; fan-out; transduction; transformations; constraints; multiple-output; multi-level networks; NOR (NAND) network synthesis.			
l. Identifiers/Open-Ended Terms			
1. COSATI Field/Group			
Availability Statement		19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 83
		20. Security Class (This Page) UNCLASSIFIED	22. Price

DEC 17 1974



UNIVERSITY OF ILLINOIS-URBANA



3 0112 001932463